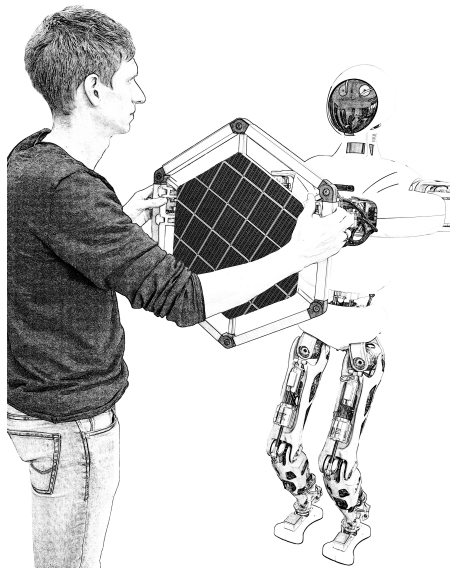


---

# Learning Task Constraints for Whole-Body Control of Robotic Systems

---

von Dennis Mronga



DISSERTATION

ZUR ERLANGUNG DES GRADES EINES DOKTORS DER  
INGENIEURWISSENSCHAFTEN  
- DR.-ING. -

VORGELEGT IM FACHBEREICH 3 (MATHEMATIK & INFORMATIK)  
DER UNIVERSITÄT BREMEN

May 19, 2022

Datum des Promotionskolloquiums: 11. Mai 2022

Gutachter Prof. Dr. Dr. h.c. Frank Kirchner  
Prof. Dr.-Ing. Udo Frese



---

# Abstract

---

Over the past two decades Whole-Body Control (WBC) has become the standard method for controlling robots with redundant degrees of freedom, such as humanoids or mobile manipulators. WBC enables the simultaneous execution of multiple tasks by formulating them as constraints or within the cost function of an instantaneous optimization problem. In each control cycle, the optimization problem is updated, solved and its solution is applied to the robot's actuators. Instead of computing the inverse kinematics or inverse dynamics of each task individually, WBC determines the optimal solution that considers all tasks, as well as physical constraints such as contact forces or actuator limits. In this way, complex control problems can be designed by combining simple tasks and the full degrees of freedom of the robot can be exploited.

However, it requires a lot of expertise to model the optimization problem in such a way that the desired robot behavior is achieved. A human expert must analyze the task, derive appropriate task models, define constraints, and assign suitable priorities to the tasks. This process is commonly performed by hand, which is time-consuming and prone to errors. Moreover, the solutions developed are usually limited to certain situations. If the given task or the environment of the robot changes, these manually designed solutions may fail and the task specification must be adapted.

In this thesis, we address these very problems to improve usability, adaptability, and generality of existing WBC approaches. First, we introduce a programming by demonstration (PbD) approach for whole-body controllers. The approach derives a part of the optimization problem, namely the task constraints and their associated priorities, from user demonstrations. The demonstrations are performed in varying conditions, which we refer to as contexts. Using the acquired data, we can derive probabilistic models that allow generalization of task constraints, and their associated priorities with respect to novel, previously unseen contexts. That is, the whole-body controller learns to adapt to unknown situations. The proposed method not only significantly reduces the effort required to design the optimization problem, but it also improves the performance of the robot in dynamic environments. Furthermore, as the approach automatically adapts the whole-body controller to the current situation, it can be used to keep task descriptions in WBC general and abstract. Thus, the approach has the potential to bridge the gap between numerical task specifications and higher-level concepts such as symbolic task planning. As a second contribution we present different methods for black-box optimization of task priorities, which may increase the performance of the derived whole-body controller when deployed on the target robot. Third, we integrate these contributions in a modular Whole-Body Control framework named ARC-OPT, which allows us to automatically derive, adapt, and optimize whole-body behaviors, while preserving the positive features of classical WBC approaches.



---

# Zusammenfassung

---

In den vergangenen zwei Jahrzehnten hat sich Whole-Body Control (WBC) als Standardmethode zur Regelung von Robotern mit redundanten Freiheitsgraden wie zum Beispiel Humanoiden oder mobilen Manipulatoren etabliert. WBC ermöglicht die gleichzeitige Ausführung mehrerer Aufgaben, indem es diese als Nebenbedingungen oder innerhalb der Kostenfunktion eines Online-Optimierungsproblems definiert. In jedem Regelzyklus wird das Optimierungsproblem aktualisiert, gelöst und seine Lösung als Steuersignal auf die Aktuatoren des Roboters abgebildet. Anstatt die Inverse Kinematik oder Dynamik jeder Aufgabe einzeln zu berechnen, bestimmt WBC die optimale Lösung, welche alle Aufgaben, sowie physikalischen Nebenbedingungen wie zum Beispiel Kontaktkräfte oder Gelenkgrenzen berücksichtigt. Auf diese Weise können komplexe Roboteraufgaben aus einfacheren Teilaufgaben entworfen und die gesamten Freiheitsgrade des Roboters optimal genutzt werden.

Allerdings wird Expertenwissen benötigt, um das Optimierungsproblem so zu modellieren, dass das gewünschte Roboterverhalten erzielt wird. So ist es notwendig die Aufgabe zu analysieren, entsprechende Aufgabenmodelle abzuleiten, Nebenbedingungen zu definieren und den Aufgaben geeignete Prioritäten zuzuweisen. Dieser, meist manuell ausgeführte Vorgang, ist sehr zeitaufwendig und fehleranfällig. Darüber hinaus sind die entwickelten Lösungen meist auf bestimmte Situationen beschränkt. Falls sich die Aufgabe oder die Umgebung des Roboters verändert, schlagen die händisch entwickelten Lösungen meist fehl und die Aufgabenbeschreibung muss angepasst werden.

In dieser Dissertation werden diese Probleme adressiert um die Benutzbarkeit, Anpassungsfähigkeit und Allgemeingültigkeit existierender WBC Ansätze zu verbessern. Erstens wird hier ein Programmierung-durch-Vormachen-Ansatz für Whole-Body Controller vorgestellt. In dem Ansatz wird ein Teil des Optimierungsproblems, nämlich die Aufgabenbeschränkungen, sowie die zugehörigen Aufgabenprioritäten aus Benutzerdemonstrationen abgeleitet. Die Benutzerdemonstrationen werden in verschiedenen Bedingungen durchgeführt, die hier als Kontexte bezeichnet werden. Anhand der akquirierten Daten werden wahrscheinlichkeitstheoretische Modelle abgeleitet, die eine Generalisierung der Aufgabenbeschränkungen und Prioritäten bezüglich neuer, unbekannter Kontexte ermöglichen. Das heißt, der Whole-Body Controller lernt, sich an unbekannte Situationen anzupassen. Der vorgestellte Ansatz hilft zum einen dabei, den Aufwand für den Entwurf des Optimierungsproblems deutlich zu reduzieren. Zum anderen verbessert er die Leistungsfähigkeit eines Robotersystems beim Einsatz in dynamischen Umgebungen. Darüber hinaus können mit Hilfe dieses Ansatzes Aufgabenbeschreibungen in WBC allgemein und abstrakt gehalten werden, da der Whole-Body Controller sich automatisch an die gegebene Situation anpasst. Demnach existiert hier das Potenzial die Lücke zwischen numerischen Aufgabenbeschreibungen und höheren Konzepten wie zum Beispiel symbolischer Aufgabenplanung zu schließen. Als zweiter Beitrag werden verschiedene Methoden zur Optimierung von Aufgabenprioritäten vorgestellt, welche die

Performance des abgeleiteten Whole-Body Controllers verbessern, wenn er auf dem Zielsystem eingesetzt wird. Drittens wird ein modulares Whole-Body Control Framework namens ARC-Opt vorgestellt, welches die zwei zuvor genannten Beiträge integriert. Das Framework ermöglicht es, Whole-Body Verhalten abzuleiten, zu adaptieren und zu optimieren, während die positiven Eigenschaften klassischer WBC Frameworks erhalten bleiben.

---

# Acknowledgments

---

This thesis would never exist if it were not for the many people who have supported, encouraged, and guided me throughout the years. I would like to express my thanks and appreciation to these people, and apologize to anyone I have forgotten.

First and foremost, I would like to thank my supervisor Professor Frank Kirchner, who gave me the opportunity to do robotics research in a unique environment at the Robotics Innovation Center of DFKI. His vision of robotics motivated me to see the big picture and not get lost in technical details. Besides him, there are many people at DFKI who influenced this work, be it by means of technical & administrative support or through all the discussions in the coffee corner and over an evening beer on a North Sea island. In this regard, I would like to thank the members of the mechanics & electronics team at RIC. They designed and constructed most of the cool robots that I got to use for experimentation over the years (AILA, Mantis, RH5, to only name a few) and they fixed whatever I broke. Also, I would like to thank all the participants of the PhD-retreats for their constructive criticism. Special thanks goes to Shivesh Kumar, Alexander Fabisch and Florian Cordes for reviewing my papers and thesis, but also for the many discussions, recommendations and hints they gave me during the last years.

In particular, I would like to thank my long-time mentor, project and team leader, and good friend Dr. José de Gea Fernández for many productive discussions and advice. José has led almost every research project I have been involved in since joining DFKI. He sparked my interest in Whole-Body Control, discussed with me many times the weaknesses of WBC approaches and ways to overcome them.

On personal level, I would like to express utmost gratitude to my parents. They were always the safe haven for me. They were on my side even when I took the wrong decisions. They allowed me an education that they themselves could not achieve, and without which I would not be where I am today.

However, my greatest and most sincere gratitude goes to my wife Yvonne Mronga. Without her love and her continuous and persistent encouragement, this work would certainly not have been possible. This dissertation is therefore dedicated to her and to our wonderful children Sophie and Jakob.



---

# Danksagung

---

Ich möchte all jenen meinen Dank und meine Anerkennung aussprechen, die mich im Laufe der Jahre unterstützt, ermutigt und angeleitet haben, denn ohne sie würde es diese Dissertation nicht geben. Außerdem entschuldige ich mich vorsorglich bei allen, die ich vergessen habe.

Zuallererst möchte ich meinem Doktorvater Professor Frank Kirchner danken, der mir die Möglichkeit gab, in einem einzigartigen Umfeld am Robotics Innovation Center des DFKI zu forschen. Seine Vision der Robotik brachte mich dazu, das große Ganze zu sehen und mich nicht in technischen Details zu verlieren. Neben ihm gibt es viele Menschen am DFKI, die diese Arbeit beeinflusst haben, sei es durch technische & administrative Unterstützung oder durch all die Diskussionen in der Kaffecke und bei einem abendlichen Bier auf einer Nordseeinsel. In diesem Zusammenhang möchte ich den Mitgliedern des Mechanik- und Elektronikteams am RIC danken. Sie haben die meisten der großartigen Roboter entworfen und gebaut, die ich im Laufe der Jahre zum Experimentieren benutzen durfte (AILA, Mantis, RH5, um nur einige zu nennen), und sie haben ohne zu klagen alles repariert, was ich kaputt gemacht habe. Außerdem möchte ich mich bei allen Teilnehmenden der PhD-Retreats für ihre konstruktive Kritik bedanken. Besonderer Dank geht an Shivesh Kumar, Alexander Fabisch und Florian Cordes für die Begutachtung meiner Publikationen und Dissertation, aber auch für die vielen Diskussionen, Empfehlungen und Hinweise, die sie mir in den letzten Jahren gegeben haben.

Im Besonderen möchte ich meinem langjährigen Mentor, Projekt- und Teamleiter und guten Freund Dr. José de Gea Fernández für viele produktive Diskussionen und Ratschläge danken. José hat fast jedes Forschungsprojekt geleitet, an dem ich seit meinem Eintritt ins DFKI beteiligt war. Er weckte mein Interesse am Thema Whole-Body Control, diskutierte mit mir viele Male die Schwächen von WBC Ansätzen und Wege, sie zu überwinden.

Auf persönlicher Ebene möchte ich meinen Eltern größte Dankbarkeit aussprechen. Sie waren immer der sichere Hafen für mich. Sie waren auf meiner Seite, auch wenn ich falsche Entscheidungen getroffen habe. Sie haben mir eine Ausbildung ermöglicht, die sie selbst nicht erreichen konnten, und ohne die ich heute sicherlich nicht dort wäre, wo ich bin.

Mein größter und aufrichtigster Dank gilt jedoch meiner Frau, Yvonne Mronga. Ohne ihre Liebe, und ihren anhaltenden und beharrlichen Zuspruch wäre diese Arbeit sicherlich nicht möglich gewesen. Diese Dissertation ist daher ihr gewidmet, und unseren wunderbaren Kindern Sophie und Jakob.





---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Goals of this Thesis . . . . .	4
1.3	Terminology . . . . .	5
1.4	Thesis Structure and Contributions . . . . .	6
1.4.1	Own Publications . . . . .	7
<b>2</b>	<b>Foundations and Background</b>	<b>9</b>
2.1	Whole-Body Control - Principles & Approaches . . . . .	9
2.1.1	Redundant Robots . . . . .	11
2.1.2	Closed-Form WBC Approaches . . . . .	13
2.1.3	Optimization-Based WBC Approaches . . . . .	17
2.1.4	Task Models . . . . .	20
2.1.5	Task Prioritization . . . . .	23
2.1.6	Floating Base Systems . . . . .	24
2.2	Programming-by-Demonstration . . . . .	25
2.2.1	User Demonstrations . . . . .	26
2.2.2	Skill Representation and Generalization . . . . .	26
2.3	Discussion . . . . .	31
<b>3</b>	<b>State of the Art</b>	<b>33</b>
3.1	PbD-Based Approaches . . . . .	34
3.2	Optimization- and RF-Based Approaches . . . . .	37
3.3	Knowledge-Based Approaches . . . . .	38
3.4	Other Approaches . . . . .	40
3.5	Relation to Behavior Learning . . . . .	40
3.6	Discussion and Summary . . . . .	40
<b>4</b>	<b>Learning Context-Adaptive Task Constraints from Demonstration</b>	<b>43</b>
4.1	Motivation . . . . .	43
4.2	Approach . . . . .	44
4.3	WBC Framework and Task Constraints . . . . .	46
4.4	User Demonstrations . . . . .	48
4.4.1	Representation of Context . . . . .	48
4.4.2	Representation of Task Constraints . . . . .	49
4.5	Estimation of Task Constraints with Soft Priorities . . . . .	50
4.5.1	Probabilistic Encoding of Context and Constraints . . . . .	51
4.5.2	Reproduction of Task Constraints . . . . .	52

4.5.3	Results	54
4.5.4	Comparison of Different Approaches for Task Prioritization	61
4.6	Alternative Encoding: Probabilistic Movement Primitives	63
4.6.1	Probabilistic Encoding and Reproduction of Task Constraints	63
4.6.2	Results	64
4.7	Discussion	69
<b>5</b>	<b>Optimization of Task Priorities</b>	<b>71</b>
5.1	Optimizing Task Constraints in Human-Robot Collision Avoidance	71
5.1.1	Motivation	71
5.1.2	Related Work: Collision Detection and Avoidance	72
5.1.3	Robot-Obstacle Distance Computation	73
5.1.4	Task-Compliant Collision Avoidance using WBC	74
5.1.5	Optimization of WBC Parameters	76
5.1.6	Results	77
5.2	Optimization of Soft Task Priority Functions	83
5.2.1	Motivation	83
5.2.2	Approach	84
5.2.3	Results	85
5.3	Discussion	88
<b>6</b>	<b>ARC-OPT: An Adaptive Whole-Body Control Framework</b>	<b>89</b>
6.1	Motivation	89
6.2	Framework Overview	91
6.2.1	WBC Library	91
6.2.2	Rock Integration	95
6.2.3	Learning Module and PbD Pipeline	98
6.3	Application: WBC for Series-Parallel Hybrid Robots	99
6.3.1	Motivation	99
6.3.2	Constrained Kinematics and Dynamics	102
6.3.3	WBC Approach	102
6.3.4	Results	104
6.4	Discussion	111
<b>7</b>	<b>Conclusion</b>	<b>113</b>
7.1	Summary and Discussion	113
7.2	Future Work	114
	<b>APPENDIX</b>	<b>117</b>
<b>A</b>	<b>Kinematics of Open Chains</b>	<b>119</b>
A.1	Differential Kinematics	119
A.2	Singularities	120
A.3	Inverse Differential Kinematics	121
A.4	Singularity Robustness	121

<b>B</b>	<b>Dynamics of Open Chains</b>	<b>123</b>
<b>C</b>	<b>Linear Least Squares and Quadratic Programming</b>	<b>125</b>
<b>D</b>	<b>Relation between WBC and Optimal Control</b>	<b>127</b>
<b>E</b>	<b>Results on Optimizing Task Priority Functions</b>	<b>129</b>
	<b>Notations and Symbols</b>	<b>133</b>
	<b>List of Figures</b>	<b>137</b>
	<b>List of Tables</b>	<b>139</b>
	<b>Glossary</b>	<b>141</b>



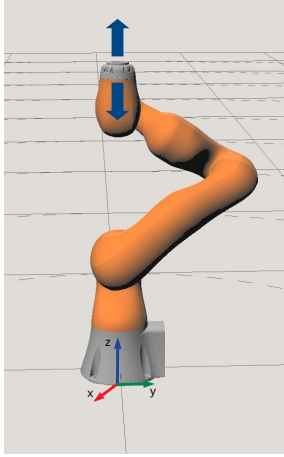
This chapter provides the motivation and scope of this thesis by shortly summarizing the area of Whole-Body Control, its limitations and the employed methods to overcome them. Furthermore, it introduces the terminology commonly used throughout this thesis and describes the structure of this document alongside the main scientific contributions.

## 1.1 Motivation

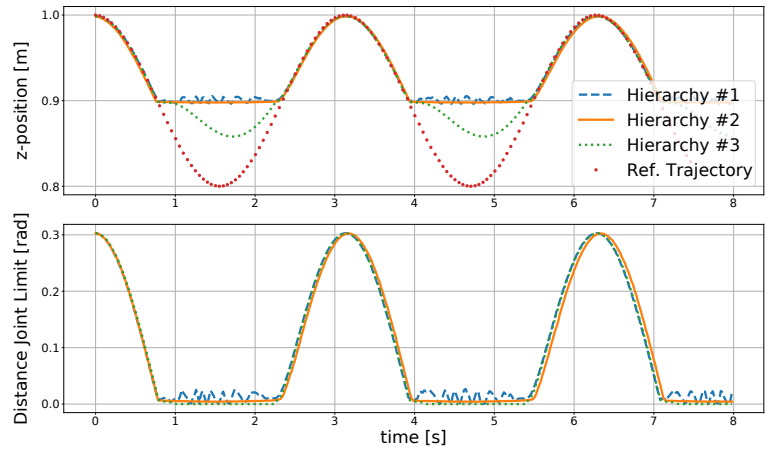
During the past two decades an increasing number of humanoid robots and other complex robotic systems like multi-legged walking machines and mobile manipulators with a single or two arms have become physically available. With the growing availability of these systems there was also an increasing need for control approaches that facilitate their deployment in human environments. These approaches should allow the integration of multiple behaviors like interaction with several contact points, manipulation, balance, and locomotion, while considering physical constraints and efficiently coordinating all degrees of freedom (dof). Until today, Whole-Body Control (WBC) has become the most widely adopted approach for such multi-objective control problems in robotics [MS19]. The core principle of WBC is to describe robot tasks as constraints or within the cost function of an optimization problem, typically a quadratic program (QP). In every control cycle, the optimization problem is updated with the current system state and its solution is applied to the robot's actuators. The solution represents the instantaneous control signal that best accomplishes all tasks simultaneously. In order to resolve conflicting tasks or emphasize the importance of different objectives, task priorities can be assigned. Depending on the prioritization method, these are referred to as strict or soft task priorities. Strict prioritization approaches establish a task hierarchy such that tasks of lower priority do not disturb higher prioritized tasks. When using soft task priorities, the solution is a weighted combination of the solutions of the individual tasks. Both prioritization schemes have their benefits and choosing the wrong task priorities may produce undesirable results, a fact that will be elaborated in detail within this thesis.

WBC has been successfully applied to complex robot control problems, such as jumping [Bel+18], climbing stairs [CKT19] or ladders [Vai+15], as well as various mobile manipulation tasks [Die+12b; Lei+16]. However, it requires a lot of expertise to model the optimization problem in a way that the desired robot behavior is achieved. A human expert must analyze the task, derive task models, define constraints, and assign suitable priorities. This procedure is performed in a manual, trial-and-error fashion and it requires detailed knowledge about the problem domain, the actual WBC implementation in use and about WBC in general. Even with that background knowledge, the procedure is still time-consuming, and the resulting robot performance is often suboptimal.

Another issue of most existing works in WBC is the low generality of the developed solutions. Tasks like, e.g., opening a door, climbing a ladder or bi-manual grasping of an object are often addressed individually, that is the Whole-Body Controller is tailored to a specific task



(a) Simulation of a KUKA iiwa robot.



(b) Resulting motion for different task hierarchies. Upper: Task space motion (only z-axis), Lower: Distance to joint position limit, (only elbow joint).

Figure 1.1: Illustrative example on the effect of different task hierarchies.

and situation. Although WBC has produced impressive results on individual problems this way, these carefully handcrafted solutions will fail if the characteristics of the given task or the environment change.

In the following, we provide illustrative examples regarding both issues.

### Example 1: Manual Selection of Task Priorities

Consider a whole-body controller that integrates two tasks on the industrial robot arm shown in Figure 1.1(a), namely following a sinusoidal end-effector trajectory and avoiding joint limits. The latter exercises a repelling force on a joint that approaches a position limit. When applying a strict task hierarchy, there are three different possibilities for prioritization:

#	Task Hierarchy		
1	Joint Limits	>	Trajectory Following
2	Joint Limits	==	Trajectory Following
3	Joint Limits	<	Trajectory Following

Table 1.1: Task hierarchies used in Example 1.

Here  $<$  and  $>$  denote lower and higher priority of the task on the left side, respectively, and  $==$  denotes equal priority for both tasks. Figure 1.1(b) shows the resulting motion for the three different task hierarchies. Here, the upper figure depicts the motion in Cartesian space along the z-axis and the lower figure shows the distance to the position limit of the elbow joint. Intuitively, joint limit avoidance should be assigned the highest priority. However, the corresponding task hierarchy #1 shows strong irregularities at the activation point of the joint limit. The reason is that the higher prioritized joint limit avoidance task pushes joint 3 away from the position boundary, until the repelling force decreases and the trajectory following task pushes the robot in opposite direction again. This leads to recurrent activation and deactivation of the unilateral joint limit constraint, a problem that has been investigated in



Figure 1.2: Illustrative example on the effect of context changes. Screenshot from video [Mro21d]

the scientific literature on WBC [MKK09]. In contrast, hierarchy #2 shows smoother transition behavior. When using hierarchy #3 the behavior is also smooth, but the joint limit of the regarded joint is not properly avoided.

This example shows that even for simple problems with only two tasks, the resulting robot behavior may differ from the expected one due to incorrect prioritization. In the example, we employ a strict task hierarchy, which offers a finite number of prioritizations. In contrast, soft task priorities, also referred to as task weights, have a continuous range of values. Thus, they offer more flexibility for task prioritization, with the downside of higher complexity and effort in selecting them. It shows that there is a strong need for automatized procedures to select task priorities in WBC approaches [Mod+16b].

### Example 2: Context Change

Another crucial point is the adaptability of Whole-Body Controllers in varying situations, which we refer to as contexts. If the context changes during task execution, the tasks and their priorities may have to be adapted. As an example, consider the problem of grasping and carrying a tray with two hands. In the approach phase a coordinated arm movement is not strictly necessary. However, after the tray has been grasped, the relative position and orientation of the hands must be constrained. To achieve this, a high-priority task controlling the relative pose of the hands can be introduced the moment the tray is grasped. Now, consider carrying the empty tray around with two hands. The tray orientation with respect to the ground can be arbitrary, leaving the required robot dof free for the execution of other tasks. However, if somebody places a bottle on the tray, it must be held horizontally as shown in Figure 1.2. Thus, the priority for controlling the tray orientation must be increased before placing the bottle. In both examples, the overall robot behavior is adapted through online modification of the tasks and their priorities. However, it is difficult to estimate the optimal timing of task insertion or priority switching. Moreover, such an adaptation can hardly be programmed in advance for every situation.

This example shows that there is also a need for adaptive WBC solutions, that is methods that automatically select the correct configuration of the whole-body controller with respect to the current context.

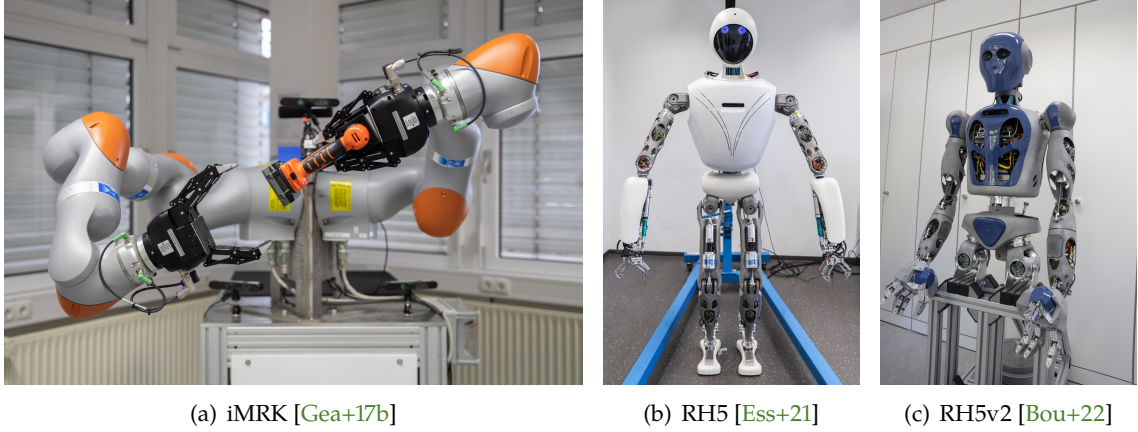


Figure 1.3: Robotic systems used for experimental evaluation in this thesis.

## 1.2 Goals of this Thesis

The general objective of this thesis is to improve the usability, adaptability, and generality of existing Whole-Body Control approaches for redundant robots. To achieve this objective, we pursue three subgoals.

- i. Introduce an automatized approach to derive task descriptions for whole-body controllers from data obtained in user demonstrations. As WBC commonly describes robot tasks as part of an optimization problem, this is equivalent to automatizing the procedure of setting up this problem. We focus on an important part of the optimization problem, namely the task constraints and their associated task priorities. Furthermore, the approach shall be able to generalize the obtained solutions to previously unseen situations (contexts) and adapt the optimization problem accordingly. The goal is to overcome the limitations of WBC as described in the previous section, namely the need of human expertise to define the WBC problem and the low generality of manually defined whole-body controllers.
- ii. Develop methods for optimization of task priorities to improve the performance of a whole-body controller when deployed on the target system. As the user demonstrations might provide suboptimal data, it is required to adapt the automatically derived WBC problem in such way that it reflects the structure of the task correctly.
- iii. Integrate the methods in a modular software framework for context-adaptive Whole-Body Control. The framework should extend existing WBC implementations with methods to intuitively specify WBC problems using PbD, adapt them to novel situations and optimize them when deployed on the target robot.

The methods not only reduce the effort of the human expert for task specification and, thus, improve usability of WBC approaches. They also increase the robot's performance and autonomy in dynamically changing environments, as the whole-body controller automatically adapts to the current context. This adaptability also allows to formulate tasks for redundant robots in an abstract and more general way. Thus, the approach has the potential to bridge the gap between numerical task specifications and high-level concepts for robot control, like symbolic task planning.



The methods developed in this thesis are evaluated on the robotic systems illustrated in Figure 1.3:

- i. **iMRK**: The iMRK system [Gea+17b] consists of two industrial KUKA LBR iiwa lightweight manipulators [AG21] and includes 14 active dof in total. The arms are equipped with Robotiq 3-finger grippers [Rob21b]. While the arms are currently arranged in a way that dual-arm manipulation is facilitated, they can be placed quite freely on the table structure. Furthermore, the system is equipped with 4 ASUS RGB-D cameras that allow detection of obstacles in the environment of the robot. The robot has a proprietary joint level impedance controller, which allows robust positioning and safe environment interaction. This controller is used in all experiments performed on the iMRK system in this thesis.
- ii. **RH5**: The humanoid robotic system RH5 [Ess+21] has been developed at the DFKI RIC. It has 32 active dof in total and is equipped with parallel grippers, also constructed at DFKI RIC. Compared to the iMRK robot it is a floating-base hybrid system, which includes multiple parallel kinematic structures. Like the iMRK system, it provides stabilizing joint level position control with compliance, which allows safe environment interaction. We use this control mode in all experiments performed on RH5 in this thesis.
- iii. **RH5v2**: The humanoid robot RH5v2 [Bou+22] has been designed as successor of the RH5 humanoid. Compared to RH5 it currently comprises only an upper body mounted on a rack. The robot has 20 active dof in total and is equipped with versatile 4-finger grippers. Like RH5 it contains multiple parallel structures as subsystems. Like the other systems, the robot provides stabilizing joint-level position control with compliance. We use this control mode in all experiments performed on RH5v2 in this thesis.

## 1.3 Terminology

**Whole-Body Control** In literature the term Whole-Body Control is often used interchangeably with the terms constraint-based control [Smi+08], optimization-based control [Fen+15], task-oriented control [SK04] or multi-objective control [DWE14]. All these terms are synonyms for the concept described in section 1.1, namely, to describe simultaneously running robot tasks as an online optimization problem, whose solution is the robot joint command that complies with all given tasks. In this thesis, we solely use the term Whole-Body Control to describe this concept.

**Task Constraints** In WBC, complex problems are described as a combination of low-dimensional descriptors in task space, for example "maintain balance", "reach object" and "avoid collision". These descriptors are mostly referred to as tasks in WBC literature, although other authors also describe them as primitives or behaviors [SK06]. In this thesis, we attempt to derive task descriptions for WBC from user demonstrations. In this regard, we are particularly interested in the constraints that the demonstrated task is subject to, and we refer to them as task constraints. If we demonstrate multiple variants of a given task, the most important task constraints correspond to the invariant features common to all demonstrations. Once extracted from the acquired data, the task constraints can be generalized to novel, previously unseen situations.

**Context** In this thesis, we use PbD approaches to derive task constraints for WBC. The task constraints shall be generalized to novel, previously unseen task variants. Throughout this thesis, the variations that a task may be subjected are referred to as contextual changes. We use context to decide, in a probabilistic manner, which task constraints are appropriate for a given task in a particular situation. As the approaches developed in this thesis are meant to adapt the robot controls with respect to context changes, they are referred to as context adaptive.

## 1.4 Thesis Structure and Contributions

The current chapter introduces to WBC and its limitations to motivate the main goals of this thesis. In Chapter 2, we provide the theoretical background on the main concepts used in this thesis, which are Whole-Body Control (WBC) and programming by demonstration (PbD). Furthermore, we present an extensive overview on existing WBC approaches and discuss their advantages and disadvantages. Chapter 3 reviews state of the art methods for automatic derivation and generalization of task constraints for WBC. The main contributions of this thesis are described in Chapters 4, 5 and 6:

- Chapter 4 presents a programming by demonstration approach to learn task constraints and their associated task priorities for whole-body controllers. User demonstrations are performed in varying conditions, which we refer to as contexts. From the acquired data, we can derive probabilistic models that generalize the derived task constraints and priorities to novel, previously unseen situations. The approach is evaluated on an industrial dual-arm robot and on a humanoid system. It increases usability, adaptability, and generality of existing WBC methods. Furthermore, we show that it provides better generalization capabilities than comparable state-of-the-art approaches for automatic derivation of whole-body controllers.
- Chapter 5 introduces approaches for black-box optimization of task priorities, which improve the performance of the derived whole-body controller when deployed on the target robot. The presented approaches are evaluated on an industrial dual-arm robot. It is shown that the developed methods improve smoothness, accuracy and dof usage compared to manually tuned whole-body controllers.
- Chapter 6 describes ARC-OPT, a modular software framework to acquire, adapt and optimize task constraints for Whole-Body Control. The framework integrates several WBC approaches on velocity, acceleration, and torque level. It provides, amongst others, an approach for modeling and solving WBC problems for series-parallel hybrid robots. The ARC-OPT framework is evaluated on two different humanoid robots.

Chapter 7 provides a summary and a discussion of the main contributions and outlook on future work.

Figure 1.4 shows an overview of the main contributions of this thesis and how they integrate. We use the ARC-OPT framework to derive task constraints from multiple user demonstrations, which are performed in different contexts. The task constraints and contexts are encoded as probabilistic models. The learned models are used to reproduce the task constraints and their associated task priorities in novel, previously unseen situations. As the solutions are deployed on the target robot, the task priorities can be improved using black-box optimization

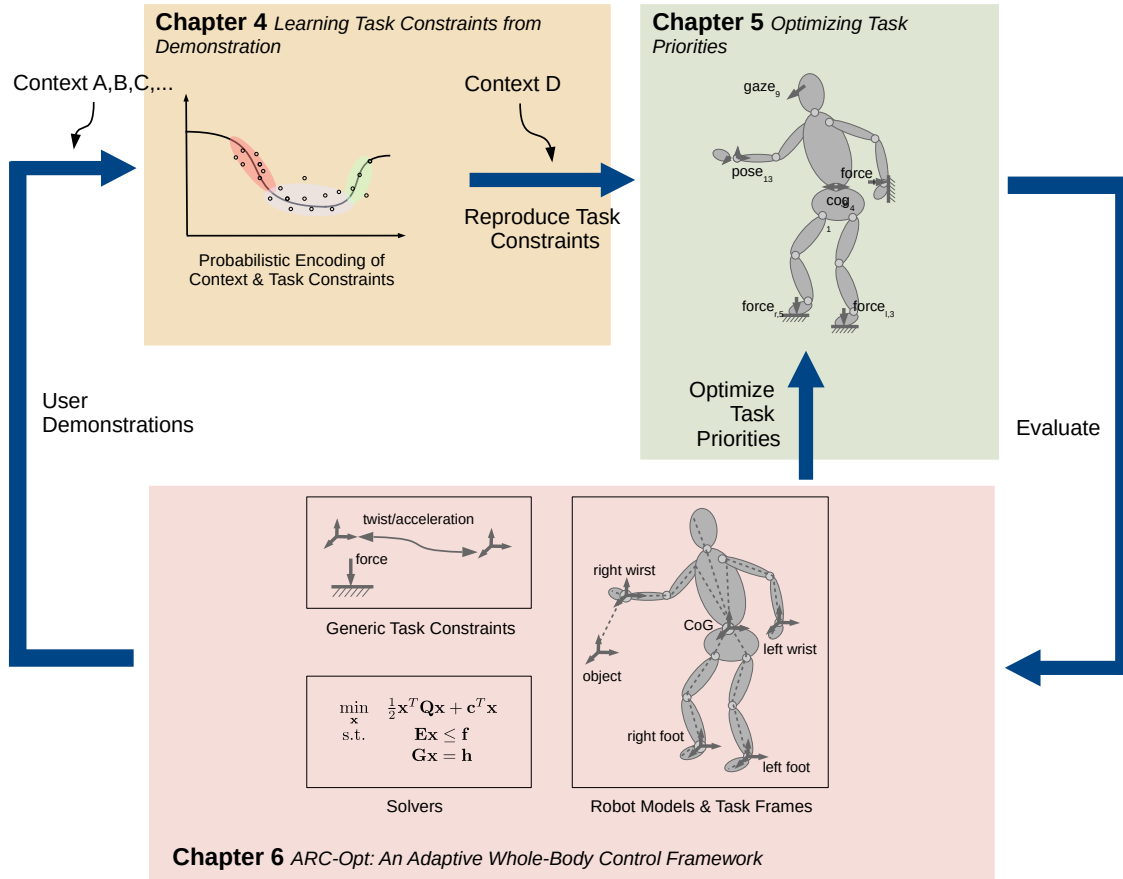


Figure 1.4: Overview of the contributions of this thesis.

methods. The fitness evaluation needed for optimization is again performed using ARC-OPT. Apart from that, ARC-OPT also serves as integrative framework for the contributions of this thesis.

### 1.4.1 Own Publications

Most of the work described in this thesis has been published before or is submitted to journals or conferences and will be published soon. The related publications are mentioned in marginal notes at the corresponding sections and summarized in the following:

#### Journals

- Dennis Mronga, Frank Kirchner: "Learning context-adaptive task constraints for robotic manipulation", In *Robotics and Autonomous Systems*, Elsevier, volume 141, 2021
- Dennis Mronga, Tobias Knobloch, José de Gea Fernández, Frank Kirchner: "A Constraint-Based Approach for Human-Robot Collision Avoidance", In *Advanced Robotics*, Taylor & Francis Online, volume 0, pages 1-17, 2020
- José de Gea Fernández, Dennis Mronga, Martin Günther, Tobias Knobloch, Malte Wirkus, Martin Schröer, Mathias Trampler, Stefan Stiene, Elsa Andrea Kirchner, Vinzenz Bargsten, Timo Bänziger, Johannes Teiwes, Thomas Krüger, Frank Kirchner: "Multi-modal Sensor-Based Whole-Body Control for Human-Robot Collaboration in Industrial

Settings", In *Robotics and Autonomous Systems*, Elsevier, volume 94, pages 102-119, 2017.

- José de Gea Fernández, Dennis Mrona, Martin Günther, Malte Wirkus, Martin Schröer, Stefan Stiene, Elsa Kirchner, Vinzenz Bargsten, Timo Bänziger, Johannes Teiwes, Thomas Krüger, Frank Kirchner: "iMRK: Demonstrator for Intelligent and Intuitive Human-Robot Collaboration in Industrial Manufacturing", In *KI - Künstliche Intelligenz*, German Journal on Artificial Intelligence - Organ des Fachbereiches "Künstliche Intelligenz" der Gesellschaft für Informatik e.V., Springer, volume 31, number 2, pages 203-207, 2017.

## Conferences

- Dennis Mrona, Shivesh Kumar, Frank Kirchner: "Whole-Body Control of Series-Parallel Hybrid Robots", 2022 IEEE International Conference on Robotics and Automation (ICRA), Accepted for publication, 2022.
- Melya Boukheddimi, Shivesh Kumar, Heiner Peters, Dennis Mrona, Rohan Budhiraja, Frank Kirchner: "Introducing RH5V2: "A Powerful Humanoid Upper Body Design for Dynamic Movements", 2022 IEEE International Conference on Robotics and Automation (ICRA), Accepted for publication, 2022.
- José de Gea Fernández, Nils Niemann, Sebastian Stock, Martin Günther, Dennis Mrona, Hendrik Wiese, Rohit Menon, Elsa Andrea Kirchner, Stefan Stiene: "Hybr-iT Project: Initial Steps Towards Contextual Robotic Manipulation for Human-Robot Teams in Industrial Environments ", Poster at the 21st International Conference of the Catalan Association for Artificial Intelligence (CCIA 2018), 2018.
- José de Gea Fernández, Dennis Mrona, Malte Wirkus, Vinzenz Bargsten, Behnam Asadi, Frank Kirchner: "Towards Describing and Deploying Whole-Body Generic Manipulation Behaviours", In 2015 Space Robotics Symposium, Glasgow, IET, University of Strathclyde, 2015.

## Poster

- Jose de Gea Fernandez, Dennis Mrona, Martin Günther, Sebastian Stock, Nils Niemann, Hendrik Wiese, Rohit Menon, Elsa Andrea Kirchner, Stefan Stiene: "Towards Contextual Robots for Collaborative Manufacturing", In Poster at the Workshop "Human-Robot Cooperation and Collaboration in Manipulation: Advancements and Challenges" at 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018), Madrid, 2018.

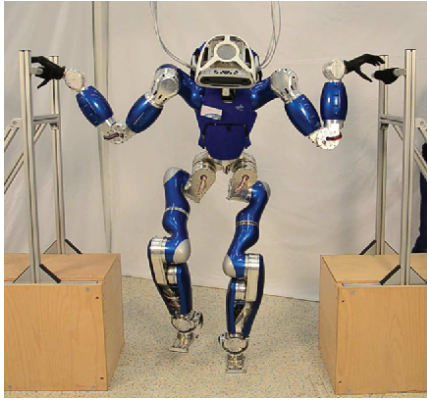
In this chapter we introduce the mathematical basics and give an overview on Whole-Body Control and programming by demonstration, which are the two core concepts used in this thesis. The mathematical foundations, in particular those in Sections 2.1.1 and 2.2.1 are mostly taken from standard robotics textbooks [LP17; Sic+08]. Apart from giving a theoretical introduction on WBC, we also provide an extensive survey on existing WBC approaches.

This chapter is organized in two main sections. Section 2.1 provides the theoretical background on WBC. After briefly recapitulating the basics on kinematics and dynamics of redundant manipulators in Section 2.1.1, we introduce closed-form and optimization-based WBC approaches on velocity, acceleration, and torque level in the Sections 2.1.2 and 2.1.3. Afterwards, we discuss different task models in Section 2.1.4, the role of prioritization in Section 2.1.5 and under-actuated systems in Section 2.1.6. Section 2.2 provides the basics of PbD. We first introduce different methods to provide user demonstrations in Section 2.2.1. Afterwards we give an overview on methods for skill representation and generalization of robot behaviors in Section 2.2.2. Finally, we provide a short discussion of the presented methods in Section 2.3.

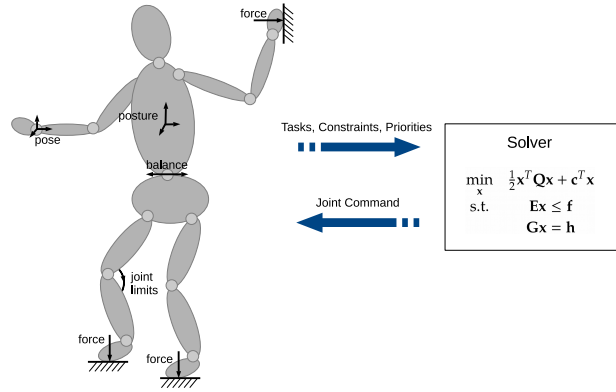
## 2.1 Whole-Body Control - Principles & Approaches

Although its theoretical foundations originate more than 40 years ago, the topic of Whole-Body Control has been intensely investigated by the robotics community mostly within the last two decades. When more humanoid robots became physical available, the need for suitable control approaches stimulated research efforts and lead to the development of the first WBC approaches. The term Whole-Body Control itself has been established by Luis Sentis in his seminal work on humanoid robot control in human environments [SK06]. WBC represents a class of feedback controllers that describe complex robot tasks as an optimization problem. It combines low-dimensional task descriptors and aggregates them into complex robot behavior, projecting the solution into the entire configuration space of the robot. However, using all the robot's degrees of freedom is not a sufficient criterion to describe a WBC approach [Soc21]. A WBC system should particularly allow the parallel execution of several tasks and integration of constraints, for example reaching for an object, while avoiding collisions and maintaining balance. Typically, tasks are described as functions to be minimized in task space and integrated as constraints or within the cost function of an instantaneous optimization problem. This problem is solved online as the robot is moving. The optimal solution is the velocity, acceleration or force/torque of the robot joints that best performs all given tasks. In every control cycle, the optimization problem is updated with the current system state. This indicates that WBC is a reactive control approach, which can be applied in dynamically changing environments. Figure 2.1 illustrates the basic principles of WBC.

Another important aspect and powerful feature of WBC approaches is the possibility to assign priorities to tasks of different importance. Priorities can be strict or soft and they can be



(a) The humanoid robot TORO performing WBC [HRO16].



(b) WBC principle.

Figure 2.1: Illustrations on Whole-Body Control

used to resolve conflicting tasks or emphasize certain aspects of the overall control problem. When using strict priorities, tasks of lower priority do not disturb higher prioritized tasks, which is a way to establish task hierarchies. When using soft task priorities, the solution is a weighted combination of the solutions of the individual tasks.

Nowadays, many different WBC algorithms, frameworks and tools exist. While they have mostly been developed with the aim of controlling humanoid robots, which are under-actuated system with a floating base, it is meaningful to apply WBC to less complex, fixed-base systems like dual-arm robots or mobile manipulators as it may simplify the task specification process. In this regard, many robotic tasks like dual-arm manipulation, wiping a window or opening a door can be described as a combination of simpler subtasks. For example, the task of wiping a window can be split into the subtasks "maintain surface contact" and "follow trajectory". Such tasks can be easily described in WBC by formulating each subtask as a function to be minimized and integrating it as a constraint or within the cost functional of the WBC problem. Obviously, maintaining surface contact is more important than following an exact trajectory along the window surface. This fact can be formalized by assigning appropriate priorities and the resulting solution will reflect the desired task hierarchy. Apart from that, every robotic manipulation task (even if executed on a simple robot arm) is inherently constrained. Constraints may arise due to the properties of the environment (e.g., the friction of a contact surface), the restrictions of the task at hand (e.g., a cup of coffee that must not be tilted) or the physical limitations of the robot (e.g., maximum joint velocities or forces/torque). In WBC, these constraints can be elegantly described and integrated into the overall problem.

The characteristics of WBC approaches are summarized as follows:

**Composition** In WBC, tasks are formulated as constraints to an instantaneous optimization problem. Complex robot behavior can be created by combining these low-dimensional task descriptors, which is usually easier than describing the overall problem at once.

**Modularity** Since task descriptions in WBC are based on elementary principles (constraints, cost functions) and commonly described in task space, these descriptions are modular. Thus, they can be transferred between robots and applied to various control problems.

**Reconfiguration** In WBC, tasks can be activated or deactivated by switching hierarchies or by blending task priority functions. Thus, the robot behavior can be adapted online in case of a changing environment or when the robot must perform a sequence of tasks.

**Redundancy** As the tasks are projected on the entire configuration space of the robot, the programmer does not have to deal with the problem of redundancy resolution. In principle, the number of the robot's dof can be arbitrary large. Some WBC approaches also allow over-constrained problems, where the number of task variables is higher than the number of robot dof.

**Reactivity** As WBC solves an online optimization problem in every control cycle, the approach is inherently reactive. Thus, it can be applied in dynamic environments, continuously integrating sensory feedback from various sources, and reacting to unforeseen events.

### 2.1.1 Redundant Robots

Since WBC approaches have been designed to control complex robots with many dof, the topic is related to the concepts of redundancy resolution and control of redundant manipulators. A redundant robot possesses more dof than required to fulfill a certain task. For example, when controlling the full end effector pose of a robot in Cartesian space, a minimum of six dof are required. Thus, a seven dof system is a typical example for a redundant robot. The presence of additional degrees of freedom allows joint motions, which do not change the position and orientation of the end effector, which means that the same task can be executed in different ways. The selection of one of these ways within the (usually infinite) number of solutions is referred to as redundancy resolution. The motivation for adding kinematic redundancy to a robot is to increase its dexterity, with the cost of additional complexity for solving the inverse kinematics and inverse dynamics problem. Numerous methods have been proposed over the years on how to resolve the redundancy of a robotic system. For example, the additional dof can be utilized to perform collision avoidance, minimize joint torques, maximize dexterity, or avoid kinematic singularities. Robots that are controlled with WBC like humanoids or other multi-legged systems are typically highly redundant, with 30 or more degrees of freedom. However, you may also encounter over-constrained problems where the number of task variables is higher than the number of robot dof.

The relationship between the joint and task space variables of an open kinematic chain can be obtained on position, velocity, and acceleration level. Since WBC mostly employs differential kinematics, we focus on the latter two. On velocity-level, the relationship between joint and task space is established via the Jacobian matrix:

$$\mathbf{v} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \quad (2.1)$$

where  $\dot{\mathbf{q}} \in \mathbb{R}^N$  is the robot joint velocity,  $\mathbf{v} \in \mathbb{R}^6$  is the spatial velocity or twist [LP17] of the end effector and  $\mathbf{J} \in \mathbb{R}^{6 \times N}$  is the geometric Jacobian matrix. Note that there is a distinction between the geometric and analytic Jacobian (see Appendix A for details). Furthermore, the Jacobian can be expressed in fixed frame coordinates (Space Jacobian) or moving coordinates (Body Jacobian). In this thesis, the symbol  $\mathbf{J}$  denotes the geometric Jacobian in fixed frame coordinates, unless specified otherwise.



The solution of the inverse kinematics problem provides the necessary joint space motion given a desired task space motion of a robotic manipulator. Closed-form WBC approaches typically require solving the inverse differential kinematics, which means solving (2.1) for  $\dot{\mathbf{q}}$  on velocity-level. If the robot is kinematically redundant ( $N > M$ ), the general solution of the first-order inverse differential kinematics problem is:

$$\dot{\mathbf{q}} = \mathbf{J}^+ \mathbf{v} + (\mathbf{I} - \mathbf{J}^+ \mathbf{J}) \dot{\mathbf{q}}_0 \quad (2.2)$$

where  $\mathbf{I} \in \mathbb{R}^{N \times N}$  is the identity matrix,  $\dot{\mathbf{q}}_0 \in \mathbb{R}^N$  an arbitrary joint space velocity and  $\mathbf{J}^+$  is the pseudoinverse of  $\mathbf{J}$  satisfying the Moore-Penrose conditions [Ben80]. The operator  $\mathbf{P} = (\mathbf{I} - \mathbf{J}^+ \mathbf{J}) \in \mathbb{R}^{N \times N}$  is the orthogonal projection on the null space of  $\mathbf{J}$ . The null space of a Jacobian  $\mathbf{J}$  is defined as the set of joint space velocities  $\dot{\mathbf{q}}_0$ , which cause zero task space velocities:  $\mathbf{J} \dot{\mathbf{q}}_0 = \mathbf{0}$ . Any joint space velocity, projected onto this null space, does not influence the solution of  $\mathbf{J}^+ \mathbf{v}$  [Sic+08].

Equation (2.2) provides the basis for establishing task hierarchies in closed-form velocity-based WBC approaches as described in Section 2.1.2. The term  $\dot{\mathbf{q}}_0$  can be used to achieve arbitrary secondary criteria. Many different secondary criteria have been investigated and applied to control redundant mechanisms over the years. Whitney [Whi69] introduces resolved motion rate control, which minimizes the kinetic energy and in addition allows to emphasize some task space coordinates, while others may be ignored. Liégeois [Lié77] presents an approach to maximize the distance to the mechanical stops of a redundant manipulator. The approach presented by Baillieul et al. [BHB84] maximizes the distance to the joint torque limits. Another important aspect is the avoidance of singularities, which can be achieved by maximizing the manipulability measure [Yos85]. Maciejewski and Klein [MK85] maximize the distance to an obstacle in the workspace of a manipulator.

The general solution of the second-order inverse differential kinematics problem is analogue to the solution on velocity-level and can be obtained by differentiating (2.2). Appendix A provides more detailed foundations on the robot Jacobian, the computation of its pseudo inverse and the inverse differential kinematics problem in general.

Considering the forces and torques that cause the motion of kinematic chains leads to the topic of robot dynamics. The dynamic equations, also referred to as equations of motion (EOM), can be expressed in joint space as:

$$\boldsymbol{\tau} = \mathbf{H}(\mathbf{q}) \ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \quad (2.3)$$

where  $\boldsymbol{\tau} \in \mathbb{R}^N$  is the vector of actuation forces and torques,  $\mathbf{H} \in \mathbb{R}^{N \times N}$  is the symmetric, positive-definite mass-inertia matrix in joint space and  $\mathbf{h} \in \mathbb{R}^N$  accounts for the effect of centripetal, Coriolis and gravitational forces in joint space. The solution of (2.3) corresponds to the inverse dynamics problem, namely computing the joint actuation torques given the system state  $\mathbf{q}, \dot{\mathbf{q}}$  and desired accelerations  $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_d$ . Conversely, computing the forward dynamics means solving (2.3) for the joint accelerations  $\ddot{\mathbf{q}}$  given the system state and actuation torques. The robot dynamic equations are typically either derived by the recursive Newton-Euler method or by the Lagrangian dynamics formulation [LP17]. Dynamic motion control of a robotic system usually requires accurate knowledge of the mass-inertia distribution reflected by  $\mathbf{H}(\mathbf{q})$ , which is sometimes hard to obtain [BGK16]. Equation (2.3) describes the



robot dynamics for free space motion. If the robot is in contact with the environment, the formulation changes to:

$$\boldsymbol{\tau} = \mathbf{H}\ddot{\mathbf{q}} + \mathbf{h} + \mathbf{J}_c^T \mathbf{f}_c \quad (2.4)$$

where  $\mathbf{J}_c \in \mathbb{R}^{6 \times N}$  is the contact Jacobian and  $\mathbf{f}_c \in \mathbb{R}^N$  the contact wrench. In dynamic WBC approaches, the equations of motion are usually considered as constraints to the underlying optimization problem (see Section 2.1.3). As they describe the physics of the robot, the computed solution must be consistent with these equations.

Similarly as (2.2) provides a general solution to the inverse differential kinematics problem for redundant systems, the general solution to the task space inverse dynamics can be written as:

$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{f}_t + (\mathbf{I} - \bar{\mathbf{J}}\mathbf{J})^T \boldsymbol{\tau}_0 \quad (2.5)$$

where  $\mathbf{f}_t \in \mathbb{R}^6$  is the wrench describing the primary task,  $\bar{\mathbf{P}} = (\mathbf{I} - \bar{\mathbf{J}}\mathbf{J}) \in \mathbb{R}^{N \times N}$  is the dynamically consistent null space projection of  $\mathbf{J}$  and  $\boldsymbol{\tau}_0 \in \mathbb{R}^N$  is a joint torque vector describing an arbitrary secondary objective. The term  $\bar{\mathbf{J}} = \mathbf{H}^{-1}\mathbf{J}^T\boldsymbol{\Lambda}$  is called dynamically consistent generalized inverse of  $\mathbf{J}$ , where  $\boldsymbol{\Lambda} \in \mathbb{R}^{M \times M}$  is the task space mass-inertia matrix. Similarly, as (2.2) provides the basis for most closed-form velocity-based WBC approaches, many closed-form torque-based approaches are based on (2.5).

### 2.1.2 Closed-Form WBC Approaches

The survey presented by Moro and Sentis [MS19] classifies WBC approaches into velocity-based versus torque-based approaches (according to the type of output), or into optimization-based versus closed-form methods (according to the type of solver). We follow that interpretation here, starting with closed-form WBC. Table 2.1 shows an overview on the existing WBC and redundancy resolution approaches, categorized accordingly.

Closed-form approaches are also referred to as analytical solutions for the WBC problem. They use sequences of projections, transpositions, inversions, or pseudo-inversions to compute the output joint velocities, accelerations or torques which fulfill the specified task constraints. The advantage of closed-form approaches is that they perform significantly faster than optimization-based approaches especially for large problems (many dof and tasks). Furthermore, they do not suffer from the problem of infeasible constraints like optimization-based solutions do. This means they will provide a solution even if the task descriptions are infeasible, which is a great advantage in real-time control. However, depending on the type of solver, they may provide suboptimal solutions, as constraints are relaxed in case of infeasibility. Also, they allow only equality constraints, while optimization-based approaches can include inequality constraints as well. As a workaround, closed-form approaches usually use potential fields [Kha85] or special inversion operators [MKK09] for tasks that can only be described by inequality constraints. For example, in the case of obstacle avoidance, a repulsive potential field controller can be smoothly activated when the robot approaches an obstacle. Table 2.2 summarizes advantages and disadvantages of closed-form versus optimization-based WBC approaches.

Framework	Output Type	Solver Type	Prioritization	References
Task Augmentation	velocity	closed-form	-	[SS88]
Task Priority Framework	velocity	closed-form	strict	[SS91]
Resolved Momentum Control	velocity	closed-form	strict	[Kaj+03]
Inverse Kinematic Architecture	velocity	closed-form	hybrid	[BB04]
iTaSC	velocity	closed-form	hybrid	[Smi+08]
Stack of Tasks	velocity	closed-form	strict	[Man+09]
Task Priority Based Redundancy Control	acceleration	closed-form	strict	[NHY87]
Saturation in the Nullspace (SNS)	acceleration	closed-form	strict	[FDK12]
Whole-Body Control	torque	closed-form	strict	[SK06]
Reactive Self-Collision Avoidance	torque	closed-form	strict	[Die+12a]
Multi-Task Compliance Control	torque	closed-form	strict	[ODA15]
Attractor-Based WBC	torque	closed-form	soft	[Mor+15]
Mixture of Controllers	torque	closed-form	soft	[DRS15]
Task Priority Framework	velocity	optimization	soft	[Kan+09]
Hierarchical Quadratic Programming	velocity	optimization	strict	[EMW14]
Dynamic Locomotion	acceleration	optimization	soft	[KPT14]
Generalized Hierarchical Control	acceleration	optimization	hybrid	[LTP16]
Momentum-Based Control	acceleration	optimization	soft	[Koo+16]
Optimization-Based Whole-Body Control	acceleration	optimization	soft	[Hop+16]
Dynamic Balance Control	torque	optimization	strict	[Col+07]
Prioritized Optimization for Task Space Control	torque	optimization	hybrid	[LH09]
Dynamic Whole-Body Motion Generation	acceleration/ torque	optimization	strict	[Saa+13]
Task Space Inverse Dynamics (TSID)	acceleration/ torque	optimization	soft	[PM16]
Optimization Based Full Body Control	acceleration/ torque	optimization	soft	[Fen+15]

Table 2.1: Overview on existing WBC and redundancy resolution approaches

Closed-form approaches	<ul style="list-style-type: none"> <li>+ Higher computational speed</li> <li>+ Handling of infeasible constraints</li> <li>– Suboptimal solutions due to constraint relaxation</li> <li>– Only equality constraints</li> </ul>
Optimization-based Approaches	<ul style="list-style-type: none"> <li>+ Handling of strict constraints without constraint relaxation</li> <li>+ Handling of inequality constraints</li> <li>– Low computational speed, especially for many dof and tasks</li> <li>– Problems with infeasible constraints</li> </ul>

Table 2.2: General advantages and disadvantages of closed-form and optimization-based WBC approaches

### Velocity-Based Approaches

Closed-Form, velocity-based approaches usually apply first-order differential kinematics as in (2.2) to compute the joint velocities that comply with the given tasks. The advantage over acceleration- or torque-based implementations is that they have lower computational complexity, are easier to implement and provide a more stable solution. Furthermore, especially industrial manipulators often provide only position interfaces, which complicates the application of torque-based WBC approaches. On the negative side, velocity-based approaches are not well suited for tasks including dynamic robot behavior and environment contacts as they cannot consider force and acceleration constraints.

The solution of closed-form, velocity-based WBC approaches extends the inverse differential kinematics problem in (2.5) to  $P$  tasks that are executed in parallel. The simplest way to do this is to stack Jacobians and task space velocities into one large matrix/vector and compute the joint velocities using pseudo inversion. This method is referred to as augmented Jacobian approach [SS88], which considers all tasks with equal importance. If tasks are conflicting, a residual error will remain for all conflicting tasks, which is not acceptable in most applications. To overcome the issue of task conflicts, it is required to prioritize one task over the other.

Equation (2.2) can be used to compute the joint space velocities that comply with a given task along with a secondary objective. Thereby, the applied null space projection ensures that tracking the secondary objective does not disturb the execution of the primary task. This is commonly referred to as task hierarchy or strict task prioritization. The concept can be generalized to multiple tasks using the following recursion:

$$\dot{\mathbf{q}}_i = \dot{\mathbf{q}}_{i-1} + (\mathbf{J}_i \mathbf{P}_{i-1})^+ (\mathbf{v}_i - \mathbf{J}_i \dot{\mathbf{q}}_{i-1}) \quad (2.6)$$

where  $\mathbf{P}_i$  is the null space projection for the  $i$ -th priority level and  $\mathbf{v}_i$  the desired task space velocity of the  $i$ -th task. This joint velocity solution allows to execute task  $i$  in the combined null space of the previous  $i - 1$  tasks, assigning tasks with higher index a lower priority. The term  $\mathbf{J}_i \dot{\mathbf{q}}_{i-1}$  reduces the desired task space velocity  $\mathbf{v}_i$  to compensate for the parts of the solution that have already been met on higher priority levels. For a single task, the equation simplifies to the first-order inverse differential kinematics problem in (2.2) without null space

term. In practice, the pseudo inversion of the Jacobians is not performed directly, but by using singular value decomposition as described in Appendix A.2, which is computationally less demanding for large Jacobian matrices and robust with respect to singularities.

The framework introduced by Siciliano and Slotine [SS91] provides an early implementation of the recursion in (2.6), which is sometimes referred to as stack of tasks. It establishes a task hierarchy on velocity level, where the Jacobian matrix is replaced by the augmented Jacobian [SS88]. Thus, the approach provides a combined solution of strict task priorities and task augmentation. Like most of the early approaches in the '80 and beginning of the '90, this method has been evaluated on a seven dof industrial robot arm. Many variations and extensions followed this approach, for example in order to implement smooth task sequences [MC07; AL15], integrate unilateral constraints [MKK09], or facilitate new applications like human-robot collaboration [Man+09]. Kajita et al. [Kaj+03] apply a strict prioritization scheme that realizes a desired linear and angular body momentum of a humanoid robot and establishes constraints on the foot contact points. This seminal work, referred to as Resolved Momentum Control, is the first to apply WBC to an actual humanoid robot.

The strict prioritization scheme described in (2.6) may be too restrictive in actual implementations. The reason is that tasks with high priorities may completely constrain most or all joints of the robot, leaving too few dof for the lower prioritized tasks. A less restrictive approach can be obtained by using weighting matrices:

$$\dot{\mathbf{q}} = \sum_{i=1}^P (\mathbf{W}_i \mathbf{J}_i)^+ \mathbf{v}_i = \sum_{i=1}^P \mathbf{J}_{W,i}^+ \mathbf{v}_i \quad (2.7)$$

where  $\mathbf{W}_i \in \mathbb{R}^{M \times M}$  is a diagonal weighting matrix, containing the task weights  $\mathbf{w}_i$  and  $\mathbf{J}_{W,i}$  is the weighted Jacobian of the  $i$ -th task. The task weights are often referred to as soft task priorities. They can be used to balance the contribution of individual task space variables to the solution. A disadvantage of soft prioritization is that selection of suitable task weights is difficult, and obtaining the desired overall robot behavior requires time-consuming, manual tuning.

By replacing the Jacobians in (2.6) with the weighted Jacobian from (2.7), we obtain a so-called hybrid prioritization scheme. The hybrid approach combines the advantages of soft and strict prioritization schemes. An implementation of a hybrid approach has been presented by Baerlocher and Boulic [BB04]. This approach integrates different algorithms for strict task prioritization, task weighting and singularity robustness. The approach is evaluated by means of a human computer animation. Another prominent work is the iTaSC framework introduced by Smits et al. [Smi+08]. It combines well-adopted control approaches in a coherent framework and provides the user a means to easily specify new tasks by imposing constraints on the relative motion of two rigid bodies. In addition, it allows the estimation of geometric uncertainty, which makes it particularly well suited for sensor-based tasks. The approach has been applied amongst others to a human-robot co-manipulation task [Sch+07]. Due to its ease of use and open-source implementation the framework became widely adopted and many implementation variants and extensions have been published [Dec+09; DBD13; BAS14; BD14].

## Acceleration and Torque-Based Approaches

The acceleration-level equivalents of the approaches defined in the previous section can be obtained by differentiating the respective equations. For example, differentiating (2.6) provides an approach with a strict task hierarchy on acceleration-level. An early attempt to build a general framework for multi-task prioritized control on acceleration level is reported by Nakamura et al. [NHY87]. The authors establish a task hierarchy and evaluate it by integrating obstacle avoidance with a positioning task on a seven dof robot. A modern approach worth mentioning here is the SNS (Saturation in the Nullspace) framework introduced by Flacco et al. [FDK12]. In this work the authors introduce an approach to the inverse differential kinematics problem in the presence of hard joint constraints (position, velocity, and acceleration limits). They solve the problem on acceleration level to avoid discontinuities in the commanded joint velocity and introduce task scaling to ensure feasibility.

On torque level, the seminal work from Sentis and Khatib [SK06] introduces the Whole-Body Control framework for humanoid robots. The authors establish a task hierarchy for multiple dynamic control objectives, which they classify as constraint primitives (contacts, joint limits, collision avoidance, balance, ...), task primitives (hand control, foot control, ...) and posture primitives (hip height, body posture, ...) and integrate them into one coherent solution. This WBC framework assumes a fixed hierarchy of physical constraints, task primitives and posture primitives. The general solution for prioritized, torque-based WBC can be computed as:

$$\tau_P = \sum_{i=1}^P (\mathbf{J}_i \bar{\mathbf{P}}_i^*)^T \mathbf{f}_i, \quad \mathbf{P}_0 = \mathbf{I} \quad (2.8)$$

where  $\bar{\mathbf{P}}_i^* = \bar{\mathbf{P}}_{i-1} \bar{\mathbf{P}}_{i-2} \dots \bar{\mathbf{P}}_1$  is the combined null space for the  $i$ -th priority level. Applying the joint torque  $\tau_P$  will not affect the execution of the tasks  $1 \dots P-1$ , which are higher prioritized.

Again, this strictly hierarchical scheme may be too restrictive in practical applications. Approaches based on task weighting allow for greater flexibility, for example an approach presented by Dehio et al. [DRS15]:

$$\tau = \sum_{i=1}^P w_i \tau_i \quad (2.9)$$

where  $\tau_i$  is the control torque for the  $i$ -th task representing a certain objective, e.g., minimum effort, centroidal linear and angular momentum or end effector pose. The approach is validated on a simulated humanoid. A similar approach is used by Moro et al. [Mor+15] who propose an attractor-based WBC framework.

### 2.1.3 Optimization-Based WBC Approaches

While early WBC approaches were mostly based on closed-form solutions as described in the previous section, most modern methods use numerical optimization. In contrast to

closed-form WBC approaches, optimization-based methods formulate whole-body behaviors as constraints or as part of the cost function of an instantaneous optimization problem, typically a quadratic program (QP). The general form of a QP is:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A} \mathbf{x} \leq \mathbf{b} \end{aligned} \quad (2.10)$$

where  $\mathbf{x} \in \mathbb{R}^N$  are the optimization variables,  $\mathbf{Q} \in \mathbb{R}^{N \times N}$  a symmetric matrix,  $\mathbf{c} \in \mathbb{R}^N$  a gradient vector and  $\mathbf{A} \mathbf{x} \leq \mathbf{b}$  a set of equality or inequality constraints. The optimization variable can be a joint velocity, acceleration, or torque. The advantage of optimization-based approaches is that they can explicitly specify inequality constraints, while closed-form approaches only allow equality constraints. Furthermore, it is possible to include other variables in the optimization problem, like contact wrenches [KPT14]. On the negative side, numerical optimization is usually slower than a closed form solution and brings problems regarding infeasibility of task constraints. Still, optimization-based WBC approaches are more widely used nowadays, especially in the context of humanoid robotics. In fact, it can be shown that any closed-form WBC can be expressed as an optimization problem [EMW14].

### Velocity-Based Approaches

The general solution of the inverse differential kinematics (2.2) can be written as unconstrained linear least squares problem (neglecting the null space term):

$$\min_{\dot{\mathbf{q}}} \quad \|\mathbf{J}\dot{\mathbf{q}} - \mathbf{v}\|_2 \quad (2.11)$$

which is a simplification of (2.10)<sup>1</sup>. The solution is the joint velocity  $\dot{\mathbf{q}}$  that minimizes the residual error of the task described by  $\mathbf{v}$ . Another possibility is to describe the robot tasks in terms of constraints:

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \|\dot{\mathbf{q}}\|_2 \\ \text{s.t.} \quad & \mathbf{J}\dot{\mathbf{q}} = \mathbf{v} \end{aligned} \quad (2.12)$$

which is an equality-constrained least-squares quadratic program. If the tasks are feasible, both solutions are equivalent. In case of infeasibility, (2.11) will provide an approximate solution that minimizes the task error, while the solution of (2.12) will fail, depending on the type of solver.

Naturally, more advanced solutions on velocity level have been proposed. Escande et al. [EMW14] extend the task priority framework [SS91] to inequality constraints and formulate

<sup>1</sup> The linear least squares problem in (2.11) can be transformed into a quadratic program, see appendix C

it as a sequence of quadratic programs as follows. On the first priority level, the following quadratic program is solved:

$$\begin{aligned} \min_{\dot{\mathbf{q}}, \epsilon_1} \quad & \|\epsilon_1\|_2 \\ \text{s.t.} \quad & \mathbf{J}_1 \dot{\mathbf{q}} \leq \mathbf{v}_1 + \epsilon_1 \end{aligned} \quad (2.13)$$

Here  $\epsilon_1$  is a vector of slack variables, which can relax the inequality constraint in case of infeasibility. By minimizing  $\epsilon_1$  an approximate solution is provided and a unique optimal value  $\epsilon_1^*$  is computed. On the second priority level, the fixed value  $\epsilon_1^*$  is inserted and the constraints for the second priority level are added:

$$\begin{aligned} \min_{\dot{\mathbf{q}}, \epsilon_2} \quad & \|\epsilon_2\|_2 \\ \text{s.t.} \quad & \mathbf{J}_1 \dot{\mathbf{q}} \leq \mathbf{v}_1 + \epsilon_1^* \\ & \mathbf{J}_2 \dot{\mathbf{q}} \leq \mathbf{v}_2 + \epsilon_2 \end{aligned} \quad (2.14)$$

The first line of constraints in (2.14) ensures that the solution  $\dot{\mathbf{q}}^*$  will not affect the first level of the hierarchy. This procedure is repeated until all levels of the hierarchy have been processed. A disadvantage of this approach is the high computational cost, as  $P$  quadratic programs must be solved for  $P$  hierarchy levels. Furthermore, each constraint is solved on its own priority level and all following ones. Different solutions are suggested in literature to speed up the process [LMH10; EMW14].

Again, strict prioritization as in (2.14) might be too restrictive. A soft prioritization scheme for optimization-based approaches can be achieved as follows:

$$\min_{\dot{\mathbf{q}}} \quad \left\| \sum_{i=1}^P \mathbf{W}_i (\mathbf{J}_i \dot{\mathbf{q}} - \mathbf{v}_i) \right\|_2 \quad (2.15)$$

where  $\mathbf{W}_1, \dots, \mathbf{W}_P$  are again diagonal matrices containing the task weights. The solution of this optimization problem is equivalent to (2.7).

### Acceleration and Torque-Based Approaches

Optimization-based approaches on acceleration or torque level are most popular in humanoid robotics, as they allow considering the robot dynamics, contact wrenches and torque limits of the robot actuators as constraints. Achieving stable joint level torque control for a humanoid is difficult though, as it depends on the availability of accurate dynamic models of the system. Thus, early approaches perform evaluations mostly in simulation [Col+07; LH09; Saa+11]. A simpler, more stable solution can be obtained by either using a velocity-based WBC, combined with a computed torque controller on joint level or by utilizing the output of a torque-based WBC as feed forward to a PD-position controller on joint level.



Many dynamic WBC approaches exist. As with the velocity-based approaches, the actual choice of cost function and constraints of the quadratic program (2.10) are problem specific. A popular approach is referred to as Task Space Inverse Dynamics (TSID) [PM16]:

$$\begin{aligned}
 \min_{\ddot{\mathbf{q}}, \mathbf{f}_c, \boldsymbol{\tau}} \quad & \|\mathbf{J}\ddot{\mathbf{q}} - \dot{\mathbf{v}}\|_2 \\
 \text{s.t.} \quad & \mathbf{J}_c^j \ddot{\mathbf{q}} = -\dot{\mathbf{J}}_c^j \dot{\mathbf{q}}, \quad \forall j \\
 & \mathbf{H}\ddot{\mathbf{q}} + \mathbf{h} + \mathbf{J}_c^T \mathbf{f}_c = \boldsymbol{\tau} \\
 & \boldsymbol{\tau}_m \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_M
 \end{aligned} \tag{2.16}$$

The decision variables are the joint accelerations  $\ddot{\mathbf{q}} \in \mathbb{R}^N$ , joint torques  $\boldsymbol{\tau} \in \mathbb{R}^N$  and contact wrenches  $\mathbf{f}_c \in \mathbb{R}^6$ . The problem is solved subject to a set of constraints. The first row assumes rigid, non-moving contact points, where  $\mathbf{J}_c^j \in \mathbb{R}^{6 \times N}$  is the contact Jacobian of the  $j$ -th rigid contact. The second row assures that the solution complies with the equations of motion, where  $\mathbf{H} \in \mathbb{R}^{N \times N}$  is the joint space inertia matrix and  $\mathbf{h} \in \mathbb{R}^N$  accounts for the Coriolis-centrifugal and gravity effects. The last row assures that the solution for  $\boldsymbol{\tau}$  is within the torque limits  $(\boldsymbol{\tau}_m, \boldsymbol{\tau}_M)$  of the actuated robot joints.

Task hierarchies can be established in a similar fashion as for velocity-based approaches, namely by introducing slack variables as in (2.14). Various solutions to this hierarchical inverse dynamics problem have been proposed in literature. For example, Escande et al. [EMW14] propose an approach to solve the task hierarchy in a single quadratic program, which is significantly faster than using iterative solvers.

In analogy to (2.15), a soft prioritization scheme can be established on acceleration level, by replacing the cost function of, e.g., (2.16) with the weighted sum of the cost of each individual task. Such an approach is, amongst others, used by Kuindersma et al. [KPT14], who weight the different task contributions in the objective function with respect to the balancing cost. They use a similar QP as in (2.16) but remove  $\boldsymbol{\tau}$  as a decision variable and model the contact constraints using a friction cone approximation.

Obviously, using a combination of strict and soft hierarchies is also possible in dynamic, optimization-based WBC approaches. Liu et al. [LTP16] introduce a generalized projector, which can handle strict and soft task constraints without the need to solve multiple quadratic programs. The resulting framework is referred to as Generalized Hierarchical Control. Feng et al. [Fen+15] also use a QP like in (2.16). However, they complement the inverse dynamics QP with an inverse kinematics solution that provides inputs for the stabilizing joint level torque controller in order to get a more stable overall behavior. Another prominent framework has been introduced by Koolen et al. [Koo+16]. The authors use the concept of centroidal momentum and the centroidal momentum matrix introduced by Orin and Goswami [OG08] to formulate a compact QP that relates task space acceleration, contact wrench constraints and the desired rate of change of centroidal momentum.

## 2.1.4 Task Models

In WBC, a task describes a control objective as a function to be minimized. A task is commonly implemented by means of a controller, whose control output is related to the gradient of



the task error. The error is thereby minimized while the robot is moving, offloading part of the global optimization problem into a closed-loop solution. The gradients of all tasks are integrated as constraints or with the cost function of an optimization problem, which computes the joint level command that achieves simultaneous execution of all tasks. Most task models in WBC are based in some way either on the task function approach [SLE91] on velocity-level or the operational space formulation [Kha87] on acceleration/torque level. The common idea of both approaches is to simplify the control problem by describing tasks in a suitable task space. This way, whole-body behaviors are easy to specify and can be transferred between different robots.

### The Task Function Approach

In velocity-based WBC approaches, tasks are mostly described using a variant of the task function approach introduced by Samson et al. [SLE91]. A task function is an arbitrary, derivable function  $\mathbf{e}(\mathbf{q})$ , which is related to the joint velocities by  $\mathbf{J}_t \dot{\mathbf{q}} = \dot{\mathbf{e}}$ , where  $\mathbf{J}_t = \delta \mathbf{e} / \delta \mathbf{q}$  is the analytical Jacobian. The task descriptions in many WBC frameworks are based in some way on the task function approach as it provides an elegant means to separate task description from robot kinematics. Task functions can be specified in Cartesian space, joint space or sensor space on velocity, acceleration, or force level. The task function approach has been applied to many multi-objective control problems in robotics, for example visual servoing [ECR92], joint limit avoidance [CM00] and singularity avoidance [MCR96].

Considering the velocity-based WBC approaches described in the previous sections, defining a task function means to provide an implementation for computing the desired control actions  $\mathbf{v} = \mathbf{v}_d$ . For Cartesian positioning tasks, a straight-forward solution is to use a simple proportional controller with feed forward term:

$$\mathbf{v}_d = \mathbf{v}_r + \mathbf{K}_p \begin{pmatrix} \mathbf{p}_r - \mathbf{p} \\ \theta \hat{\boldsymbol{\omega}}_r \end{pmatrix} \quad (2.17)$$

where  $\mathbf{v}_d \in \mathbb{R}^6$  is the desired spatial velocity, also referred to as twist,  $\mathbf{v}_r \in \mathbb{R}^6$  is the reference (feed forward) spatial velocity and  $\mathbf{K}_p \in \mathbb{R}^{6 \times 6}$  is a diagonal matrix containing 6 feedback gain constants. The vectors  $\mathbf{p}_r, \mathbf{p} \in \mathbb{R}^3$  are the reference and the actual position of the regarded robot frame. The term  $\theta \hat{\boldsymbol{\omega}}_r \in \mathbb{R}^3$  is a rotation vector denoting the difference in orientation between actual and reference pose using a suitable, singularity-free representation [LP17]. The computed control action  $\mathbf{v} = \mathbf{v}_d$  is fed into one of the velocity-based WBC frameworks described in the previous sections, e.g., (2.7). The solution of (2.7) regulates the control objective defined by (2.17) in task space. It minimizes the residual task space error, while the robot is moving.

Positioning in joint space can be performed in a similar fashion using the following controller:

$$\dot{\mathbf{q}}_d = \dot{\mathbf{q}}_r + \mathbf{K}_p(\mathbf{q}_r - \mathbf{q}) \quad (2.18)$$

where  $\dot{\mathbf{q}}_d \in \mathbb{R}^N$  is the control output,  $\dot{\mathbf{q}}_r \in \mathbb{R}^N$  the feed forward joint velocity and  $\mathbf{q}_r, \mathbf{q} \in \mathbb{R}^N$  the reference and actual joint configuration, respectively. Since  $\dot{\mathbf{q}}_d$  is defined in joint space, it can be integrated into a WBC problem by setting the task Jacobian to the identity matrix  $\mathbf{J} = \mathbf{I} \in \mathbb{R}^{N \times N}$  and  $\mathbf{v} = \dot{\mathbf{q}}_d$ .

Both (2.17) and (2.18) implement attractor tasks. In contrast, repulsive behavior as needed for collision avoidance can be achieved using potential fields. In Cartesian space, a radial repulsive potential field can be described as follows:

$$\mathbf{v}_d = \begin{pmatrix} \mathbf{K}_p \frac{\mathbf{p} - \mathbf{p}_0}{d} S(d) \\ \mathbf{0}^{3 \times 1} \end{pmatrix} \quad (2.19)$$

where  $d = \|\mathbf{p} - \mathbf{p}_0\|_2$  is the distance of the robot to the potential field center. When being applied to collision avoidance, the potential field center  $\mathbf{p}_0 \in \mathbb{R}^3$  represents the location of an obstacle. The term  $S(d)$  is a sigmoid function of the distance  $d$ . Using a sigmoid here ensures a bounded control action and smooth transition when approaching the obstacle. In contrast, using the reciprocal distance instead of a sigmoid may provide infinite control output close to the obstacle. In joint space, one-dimensional repulsive potential fields can be used for joint limit avoidance and are defined accordingly:

$$\dot{q}_{i,d} = k_p \frac{q_i - q_{i,0}}{d} S(d), \quad i = \{1, \dots, N\} \quad (2.20)$$

where  $d = |q_i - q_{i,0}|$  is the distance to the nearest joint limit  $q_{i,0}$  (upper or lower) of joint  $i$ . Again, this joint space task can be integrated into a WBC problem by setting the task Jacobian to the identity matrix  $\mathbf{J} = \mathbf{I} \in \mathbb{R}^{N \times N}$  and choosing  $\mathbf{v} = \dot{\mathbf{q}}_d = (\dot{q}_{1,d}, \dots, \dot{q}_{N,d})$ . Many robot behaviors can already be described through combinations of the task functions we discussed in this section. Nevertheless, many more task functions exist in literature in a large variety of implementations.

## The Operational Space Formulation

The operational space formulation has been introduced in the seminal work by Khatib [Kha87]. The approach strongly influenced the research in dynamic manipulation and control of redundant robots within the last decades. As with the task function approach, the idea is to specify tasks in a dedicated space, the operational space, which is chosen in a way that task specification is simplified. For example, considering environment interaction, one might want to specify tasks in a coordinate system located in the contact point of the manipulator with the environment or in the force sensor frame, while free space motions should best be specified in end effector coordinates. In contrast to the task function approach, the operational space formulation explicitly considers the manipulator dynamics and is therefore better suited for compliant environment interaction and force control.

Considering acceleration- or torque-based tasks, defining task models means implementing a controller that produces the desired control actions  $\dot{\mathbf{v}}_d$  for acceleration-based tasks or  $\boldsymbol{\tau}_d$

for force-based tasks. Within the operational space framework, motion control in Cartesian space is achieved by selecting the following spatial acceleration:

$$\dot{\mathbf{v}}_d = \dot{\mathbf{v}}_r + \mathbf{K}_d(\mathbf{v}_r - \mathbf{v}) + \mathbf{K}_p(\mathbf{x}_r - \mathbf{x}) \quad (2.21)$$

where  $\mathbf{x}_r, \mathbf{x} \in \mathbb{R}^6$  are the reference and actual pose,  $\mathbf{v}_r, \mathbf{v} \in \mathbb{R}^6$  the reference and actual twist and  $\dot{\mathbf{v}}_r \in \mathbb{R}^6$  the reference (feed forward) spatial acceleration of the controlled robot frame. The matrices  $\mathbf{K}_p, \mathbf{K}_d \in \mathbb{R}^{6 \times 6}$  are diagonal matrices with the proportional and derivative gain constants on the main diagonal. In joint space, the analogue expression is:

$$\ddot{\mathbf{q}}_d = \ddot{\mathbf{q}}_r + \mathbf{K}_d(\dot{\mathbf{q}}_r - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_r - \mathbf{q}) \quad (2.22)$$

where  $\mathbf{q}_r, \mathbf{q} \in \mathbb{R}^N$  are the reference and actual joint configuration,  $\dot{\mathbf{q}}_r, \dot{\mathbf{q}} \in \mathbb{R}^N$  the reference and actual joint velocity and  $\ddot{\mathbf{q}}_r \in \mathbb{R}^N$  the reference (feed forward) joint acceleration. Again, as  $\ddot{\mathbf{q}}_d$  is defined in joint space, the related Jacobian will be the identity matrix.

Substituting the above  $\dot{\mathbf{v}}_d$  or  $\ddot{\mathbf{q}}_d$  into one of the acceleration-based WBC frameworks will provide the joint space accelerations or forces/torques that comply with the given control objectives, i.e., minimize the task residual error.

Other task models have been proposed in literature for acceleration- and force-based tasks like dynamic obstacle avoidance [Kha85], self-collision avoidance [Die+12a] or task and joint space compliance [Die+12b].

### 2.1.5 Task Prioritization

As elaborated in the previous sections, tasks in WBC can be assigned different priorities in order to resolve conflicts or to emphasize the importance of certain objectives. For example, the balance of a humanoid robot would certainly have higher priority than reaching for an object, which again should be prioritized over the task of maintaining an upright upper body posture. In the WBC literature, different prioritization schemes are found, which can be classified into strict, soft or hybrid methods. Strict task hierarchies can be established for example via a sequence of null space projections as in (2.6) or by introducing slack variables into the optimization problem as in (2.13). Strict priorities ensure that tasks of higher priority are not disturbed by lower prioritized tasks. This means that the task with highest priority will be fulfilled if it is feasible, while tasks with lower priority are only fulfilled if the robot has enough redundant dof left. In general, strict prioritization schemes are more restrictive than soft prioritization, as they quickly consume all robot dof. As a result, important aspects of the control problem may be neglected. In contrast, soft prioritization schemes are usually implemented by weighting the different control objectives as in (2.7), (2.9) or (2.15). Here, the degree of task achievement depends on the assigned weight value. On the positive side, more simultaneously running tasks can be implemented this way and there is greater flexibility in prioritizing them. However, this flexibility comes with a larger effort of selecting appropriate weight values. Also, different tasks may disturb each other if they are executed using the same robot dof. Finally, hybrid prioritization schemes allow both strict and soft prioritization.

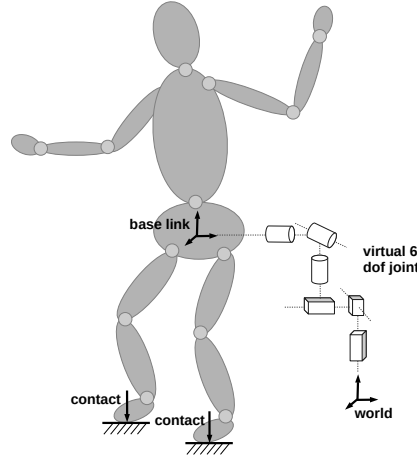


Figure 2.2: Model of a floating base robot

The role of prioritization in WBC is not limited to resolve conflicting tasks. Task priorities can also be used to influence the overall behavior of the robotic system. By switching task priorities at runtime, different robot behavior can be implemented, or task sequences can be executed [MC07]. Regarding the latter, soft task priorities can be used as activation function in order to smooth transitions between subsequent tasks. Furthermore, high-level control modules can make use of this activation/deactivation functionality to plan transitions for complex maneuvers involving such task sequences. This is a key feature of WBC regarding the development of the methods within this thesis. Through the temporal evolution of task priority functions, we adapt the robot behavior according to the current situation. We will further elaborate on this topic in Chapter 4.

### 2.1.6 Floating Base Systems

Until now we neglected the fact that most WBC approaches are designed for floating base systems, for example humanoids. The term floating base indicates that the robot is not rigidly attached to the environment, but its position and orientation with respect to the world may be changed. As the floating base state cannot be directly controlled, a humanoid is an example for an underactuated system. Underactuated means that a system cannot be commanded to follow arbitrary trajectories in configuration space, either because of physical constraints or because the system has more degrees of freedom than actuators.

A floating base is usually modeled by attaching 6 virtual joints (3 translational and 3 rotational) to the base link of the robot (see Figure 2.2). The state of this virtual linkage can be estimated from inertial sensors, force or contact sensors and the leg's forward kinematics [Xin15]. The dynamic behavior of the robot can only be controlled indirectly through the support contact points with the ground. The presence of a floating base enlarges the configuration space of the robot by 6 virtual joints as in  $\mathbf{q} = (\mathbf{x}_b \quad \mathbf{q}_u) \in \mathbb{R}^{6+N}$ , where  $\mathbf{x}_b = (x_b, y_b, z_b, \phi_b, \theta_b, \psi_b)^T$  is the pose of the robot base link in world coordinates and  $\mathbf{q}_u \in \mathbb{R}^N$  the configuration of actuated robot joints. The equations of motion (2.4) must be adapted in case of a floating base

system as follows:

$$\mathbf{S}^T \boldsymbol{\tau} = \mathbf{H} \ddot{\mathbf{q}} + \mathbf{h} + \mathbf{J}_c^T \mathbf{f}_c \quad (2.23)$$

where  $\mathbf{S} \in \mathbb{R}^{N \times (N+6)}$  is a diagonal actuation matrix, separating the actuated from the unactuated dof. Note that here the mass-inertia matrix  $\mathbf{H}$ , the bias term  $\mathbf{h}$  and the Jacobian  $\mathbf{J}$  include additional rows corresponding to the joints of the floating base.

## 2.2 Programming-by-Demonstration

Robot programming by demonstration (PbD), which is sometimes also referred to as imitation learning, is a method to specify tasks by demonstrating them to the robot, instead of explicitly programming them using machine commands. This makes PbD an intuitive approach for end users without robotics expertise to teach new robot behaviors. Apart from that, PbD may also reduce the search space for learning, either by starting from an observed (good) example or by dropping what is known as a bad solution and reducing the search space accordingly [Bil+08].

PbD has been introduced in the 1980s in the context of industrial robotics as a promising alternative to reduce time and costs for development and maintenance of robot programs in manufacturing lines. In the beginning, the demonstrated tasks consisted of simple point-to-point motions demonstrated through teleoperation. The demonstrations were recorded in joint space and directly transferred to the target robot. To consider the variability within different demonstrations from the human operator and inherent sensor noise, more sophisticated methods were required that consolidate all demonstrated movements, generalize the task, and reproduce it in a novel context. These ideas are illustrated in Figure 2.3.

Robot tasks learned by demonstration are often referred to as skills. Throughout the years PbD has been applied to learn various robotic skills, especially in combination with reinforcement learning. Examples include teaching a robot to play ball-in-a-cup [KP14] or flipping a pancake [KCC10].

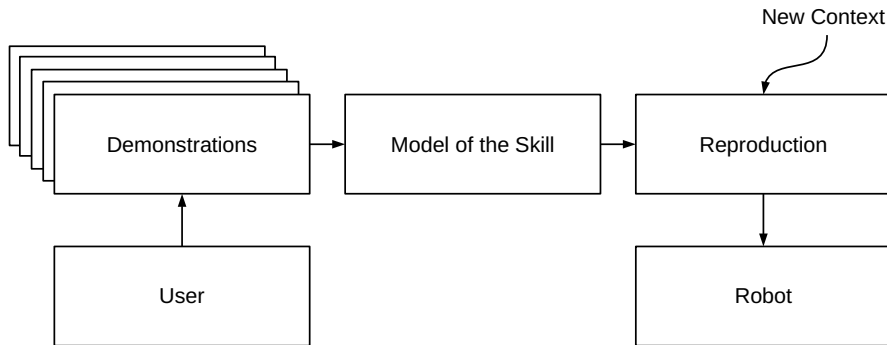


Figure 2.3: General idea of PbD, figure inspired by [Bil+08]

### 2.2.1 User Demonstrations

In PbD the data, which is used to derive the desired robot behavior, is acquired through user demonstrations. These demonstrations can be performed by several different means.

A common approach is to use kinesthetic guiding [CB08], where the robot is in a compliant or gravity mode and the operator steers the system along the desired trajectory. The advantage of this method is that the desired task can be directly captured by e.g., the proprioceptive and force sensors of the robot. This allows to demonstrate tasks with environment contacts and avoids the problem of mapping the captured motions into the workspace of the robot, which requires solving the correspondence problem [ND02]. The disadvantage is that the desired motion can be hard to perform as the robot may disallow certain motions due to kinematic singularities or other restrictions of its workspace. Furthermore, only single arm or dual arm motions can be demonstrated by a single operator. More complex behaviors like balancing or walking are hard to demonstrate through kinesthetic teaching.

Another possibility is to telemanipulate the robot either with a joystick, game controller (supported by virtual reality environment [FBB16]) or with an exoskeleton. The latter can cover a single arm, the upper body or even the full body of the operator [Kum+19c]. When using telemanipulation, a workspace mapping from input space to the kinematic structure of the robot is required. Furthermore, depending on the input device, demonstrating certain motions might not be intuitive. When using a joystick, the demonstration is usually performed in task space, which requires a certain amount of practice for the operator. When using an exoskeleton, the motions that can be demonstrated might be limited, depending on the physical workspace of the exoskeleton.

Finally, user demonstrations can be provided using a motion capture system, for example an IMU-based motion tracking suite [Cac+17] or a camera-based system [Gut+18]. Nowadays, these devices are commercially available in a large variety. Using a motion capture system is the most intuitive way of teaching skills to a robot, since it covers the entire range of motions that a human can perform. However, mapping to the robot workspace can become difficult since human and robot usually have different degrees of freedom, range of joint motion and dynamics [Pol+02]. Furthermore, interaction forces cannot be easily integrated when using motion capture systems.

### 2.2.2 Skill Representation and Generalization

Skills learned through PbD can be represented on symbolic and on trajectory level [Bil+08]. While the former method facilitates sequential organization of pre-defined skills to more complex tasks, it is limited in practical applications, where we usually want to generalize movements with respect to different task parameters, e.g., start and end position. Furthermore, it requires a predefined set of controllers that can reproduce the learned skill. Consequently, we will focus on methods that represent skills on trajectory level in this thesis. These methods may encode the skills learned from demonstrations in joint or task space, respectively.

Many different methods for representing skills learned through PbD exist. In the following we will present the ones that are most relevant for this thesis and discuss advantages and shortcomings especially with respect to their generalization capabilities in terms of learning whole-body behaviors.

## Gaussian Mixture Models and Gaussian Mixture Regression

A Gaussian Mixture Model (GMM) is a parametric probability distribution represented as a weighted combination of Gaussians:

$$p(\xi) = \sum_{k=1}^K \pi_k \mathcal{N}(\xi, \mu_k, \Sigma_k) \quad (2.24)$$

Here  $K$  is the number of mixture components,  $\pi_k$  are the mixing weights,  $\mu_k$  and  $\Sigma_k$  the means and covariances of the Gaussian distributions  $\mathcal{N}(\xi, \mu_k, \Sigma_k)$ . The modeled data is represented by the variable  $\xi$ . Note that all parameters of the model are vectors, thus (2.24) describes a multivariate probability distribution.

The classical application of GMMs is unsupervised learning of clusters within a population of normally distributed data points. For this purpose, a mixture of  $K$  Gaussians is fitted to the data without prior knowledge about what data point belongs to which cluster. Given the fitted model, a previously unseen data point can be assigned to an existing cluster. In contrast, in PbD applications GMMs are used for motion modeling and synthesis. Here, the GMM is fitted to the data obtained in  $D$  user demonstrations, where the modeled variables can be e.g., joint space or task space positions, velocities and forces/torques of the robot.

**Fitting** Fitting a GMM is mostly performed using the iterative expectation maximization algorithm [DLR77]. Starting from an initial distribution of  $K$  mixture components (randomly selected or initialized with e.g., k-means [Mac+67]) EM alternates between finding the probabilities for each data point to be generated by each mixture and fitting the mixtures to the assigned points in order to maximize the likelihood of the data given those assignments. However, selecting the optimal number of mixture components  $K$  can be difficult. One possibility is to select the number of components that maximize a certain metric like the Bayesian Information Criterion (BIC). However, this means that one has to repeat fitting multiple times for different values of  $K$ . To avoid this overhead, the selection process can be automatized by using variational inference [BKM17]. This algorithm extends EM by introducing a prior distribution on the mixing weights, which is also computed by the algorithm. Unlikely mixtures receive a low mixing weight and have no influence on the result. In practice one can simply select a large number of mixtures as an upper bound and the algorithm will distribute most of the weight on a few components, while setting all others close to zero. On the negative side, variational inference is notably slower than EM and requires additional hyper-parameters to tune. One important hyper-parameter is the prior on the weight concentration, which decides if only a few components should be emphasized, or the weights should be equally distributed over all components.

**Motion Synthesis** To retrieve a continuous motion from a GMM, the simplest way is to encode an additional time variable  $t$  together with the motion variables  $\mathbf{x}$  in the mixture model:

$$p(\xi) = p(t, \mathbf{x}) \quad (2.25)$$



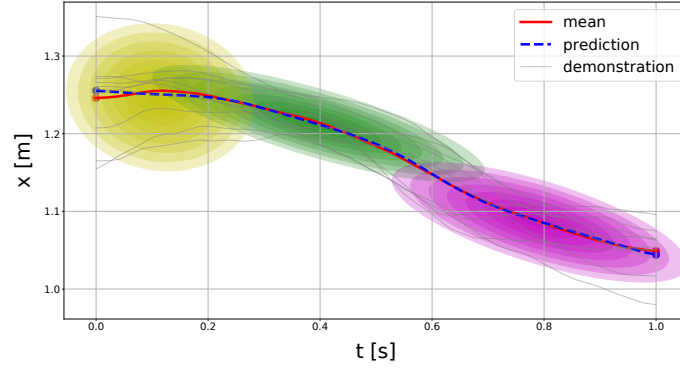


Figure 2.4: Using GMM to model a data set obtained from  $D = 10$  user demonstrations ( $K = 3$ ). GMR is applied to retrieve a continuous motion (blue dashed line).

Now, the motion variables can be reproduced for every given time step from the conditional distribution  $p(\mathbf{x}|t)$  by the means of Gaussian Mixture Regression (GMR) [CGB07]. Figure 2.4 shows an example of this procedure. Here, a GMM with  $K = 3$  mixture components is used to model a one-dimensional data set obtained from  $D = 10$  user demonstrations (grey lines). The differently colored ellipses represent the mean and spread of the respective mixture components. As can be seen, the variances of the mixtures reflect the variability in the provided user demonstrations. GMR is used to retrieve a continuous motion from the model (blue dashed line), which is close to the mean of the data set (red line).

The encoding strategy described by (2.25) is referred to as time-indexed encoding of motion. Alternatively, different subsets of input and output variables can be selected. According to [Cal16] three main usages of GMR can be observed in literature:

- i. Encoding as time-indexed trajectories. The motion is reproduced from  $p(\mathbf{x}|t)$  using GMR for each time step  $t$  (see example above).
- ii. Encoding as autonomous system. The joint probability of position  $\mathbf{x}$  and velocity  $\mathbf{v}$  of the system is learned. Starting from the current position of the system  $\mathbf{x}_0$  and time  $t = 0$ , a continuous motion is retrieved by computing velocity commands from  $p(\mathbf{v}_t|\mathbf{x}_t)$  using GMR and integrating the respective position commands  $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_t\Delta t$  until convergence to the target position
- iii. Encoding as dynamic system. The joint probability distribution of  $\xi = (\mathbf{x}, \mathbf{v}, \dot{\mathbf{v}})$  is used to generate the motion in the same manner as with the autonomous system encoding.

Each encoding has advantages and disadvantages. For example, time-indexed encoding is attractive due to its simplicity. Also, the assignment of time-indices to data points is usually unambiguous, which leads to reliable results. On the negative side, complex demonstrations with different options cannot be easily encoded with time-indexed trajectories. Furthermore, generalizing the trajectory to different starting points is not possible without further modification of the model. When encoding motion as an autonomous or dynamic system, the process of motion synthesis is initialized with the current position. Thus, generalizing with respect to different starting positions is possible by design. Also, different options in the demonstrated movements can be encoded using the latter two strategies. These advantages are achieved at the expense of increasing the model dimension by the first and second order derivatives of the motion variables.



**Generalization** Gaussian Mixture Regression can be used to generalize robotic skills obtained from PbD with respect to novel situations. In the example of time-indexed trajectory encoding, we use a time variable to obtain a continuous motion using GMR. Additionally, we can encode task parameters, which we want to generalize about, within the joint probability distribution. These parameters can be simply different starting or ending points of the demonstrated motion, but also more complex characteristics of the task such as properties of the handled objects or the environment. In the machine learning literature, the task parameters to generalize about are often referred to as *context*. Assuming we can describe the current context by a real-valued vector  $\kappa \in \mathbb{R}^C$ , where  $C$  is the number of task parameters, then the probability distribution we want to learn is  $p(t, \kappa, x)$ . From this distribution, we can retrieve a generalized version of the learned motion by applying GMR to the conditional distribution  $p(x|t, \kappa)$ . Thereby, the given context vector  $\kappa$  can describe a novel, previously unseen situation. The quality of skill reproduction in a novel context depends on the quality of the training data set and on the generalization capabilities of the model. Several approaches exist to increase the generalization capabilities of GMR, most notably the use of Task-Parameterized Gaussian Mixture Models (TP-GMMs) [Cal+12; Cal16]. Here, the task parameters are reference frames according to which a demonstrated motion is observed, each described by a linear transform comprising rotation matrix and translation vector. During reproduction, the motion can be adapted to a new frame of reference by generating a GMM using the new frame's position and orientation. Thus, the model provides some extrapolation properties that go beyond the capabilities of normal regression methods.

### Movement Primitives

Dynamic Movement Primitives (DMPs), first introduced by Ijspeert et al. [INS02], are a movement representation based on nonlinear differential equations. DMPs have been introduced with the goal of finding a robust movement representation for imitation learning in robotics. A one-dimensional DMP can be represented by the following set of differential equations [Pas+09]:

$$\begin{aligned} T\ddot{v} &= K(x_g - x) - Dv - K(x_g - x_0)s + Kf(s) \\ T\dot{x} &= v \\ T\dot{s} &= -\alpha s \end{aligned} \tag{2.26}$$

where  $x, v, \dot{v}$  are position, velocity, and acceleration of the system,  $x_0, x_g$  the start and goal position, respectively,  $K, D$  stiffness and damping factor and  $s$  is a phase variable, which monotonically changes from 1 to 0 during the movement depending on the constant factor  $\alpha$ . The equations can be interpreted as a linear spring-damper system described by the spring constant  $K$  and damping  $D$ , which is perturbed by the non-linear forcing term  $f(s)$ :

$$f(s) = \frac{\sum_i \alpha_i \Psi_i(s)s}{\sum_i \Psi_i(s)} \tag{2.27}$$

Here,  $\Psi_i(s)$  are Gaussian basis functions. The parameters of the forcing term, which converges to zero towards the end of the motion, can be learned to create the desired attractor landscape.

The parameters comprise the weights, as well as the centers and widths of the Gaussian basis functions.

DMPs have several favorable properties for movement representation in imitation learning [Pas+09]. First, convergence to the goal position  $x_g$  can be guaranteed as the forcing term vanishes towards the end of the motion. The parameters of the forcing term can be learned in order to create any trajectory shape. Furthermore, the generated motions are smooth and invariant to space and time, i.e., they can be scaled according to the desired movement duration, as well as adapted to different start and goal positions. Thus, DMPs inherently provide certain generalization abilities.

Movement primitives have a meanwhile long history in imitation and reinforcement learning. They have been successfully applied to problems like learning to play ball-in-a-cup [KP14], flipping a pancake [KCC10] or goal-directed ball-throwing [Ude+10]. Also, several variants of the original DMP formulation have been presented, which allow a non-zero target velocity [MKP10] or which can properly deal with orientation trajectories [Ude+14].

**Probabilistic Movement Primitives** In terms of learning task constraints for WBC, we are interested in the variance of the demonstrated movements in order to deduce the importance of the demonstrated task. Unfortunately, the original DMP formulation does not capture information on variance and correlation of the movement. However, Paraschos et al. [Par+13] present a DMP variant denoted as Probabilistic Movement Primitives (ProMPs), which represent a distribution over trajectories. In the given approach, a state space trajectory is represented as a linear basis function model:

$$\mathbf{y}_t = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{v}_t \end{bmatrix} = \mathbf{\Phi}_t^T \mathbf{w} + \epsilon_y \quad (2.28)$$

where  $\mathbf{\Phi}_t$  is a matrix of basis functions,  $\mathbf{w}$  a weight vector and  $\epsilon_y$  Gaussian noise. The term  $\mathbf{\Phi}_t^T \mathbf{w}$  can be used to represent the mean of a trajectory. To represent the variance of a trajectory, a probability distribution over the weight vector is introduced. The probability of observing a trajectory  $\mathcal{T}$ , given a certain weight vector can be computed as the product:

$$p(\mathcal{T} | \mathbf{w}) = \prod_t \mathcal{N}(\mathbf{y}_t | \mathbf{\Phi}_t^T \mathbf{w}, \mathbf{\Sigma}_t) \quad (2.29)$$

where  $\mathbf{\Sigma}$  is the covariance matrix of the trajectory distribution. The choice of the basis functions  $\mathbf{\Phi}$  depends on the type of movement. Paraschos et al. [Par+13] use Gaussian basis functions for stroke-based movements and Von-Mises basis functions for rhythmic motions.

ProMPs have many of the favorable properties as DMPs, like e.g., temporal scaling and spatial adaptation. Apart from that they allow a variety of probabilistic operations like conditioning of different via points, final positions or velocities, combination and blending of trajectories, and computation of variance and covariance of trajectories resulting from multiple user demonstrations. The latter is particularly important for learning task constraints as it allows us to infer task priorities by relating them to the variability of the user demonstrations.

## Other Representations

Many models for skill representation exist in robotics. Considering PbD, early approaches were focusing on Hidden Markov Models (HMMs) to encode spatial and temporal variations in user demonstrations, for example in the work introduced by Tso and Liu [TL96]. Another generic approach is to treat PbD as a regression problem and simply use a general function approximator like a neural network or Gaussian Process Regression to represent the desired skill. With the increasing popularity of Deep Learning (DL) methods in applications like facial or speech recognition, also their abilities in robotic behavior learning have been considered. DL methods usually require large data sets to achieve considerable generalization performance. Thus, training data is often generated artificially. Schmidt et al. [Sch+18] require 95000 samples acquired from simulation to train a convolutional neural network (CNN) for the purpose of unknown object grasping on a humanoid robot. In contrast, the CNN approach presented by Levine et al. [Lev+18] learns hand-eye coordination for robotic grasping from real data acquired in 800,000 grasp attempts using 14 robotic manipulators. Although many methods exist in literature to increase sample efficiency of DL methods, they still require a considerable number of training samples and therefore we do not focus on these methods within this thesis.

## 2.3 Discussion

In this chapter, we provide the theoretical background on the two core concepts used in this thesis, namely Whole-Body Control and programming by demonstration.

Regarding programming by demonstration, GMM-GMR provides a framework for motion synthesis and it has several desirable properties with respect to learning task constraints for WBC. Apart from providing smooth trajectories and good generalization capabilities with a few training samples, it provides information on the variance of the user demonstrations. This variance can be exploited to estimate the importance of certain features of the demonstrated motions, which again can be mapped to task priorities of a whole-body controller. We will elaborate on this feature in detail in Chapter 4. Compared to GMR, movement primitives provide a different motion model and properties like inherent stability and scaling according to the motion duration, start and end position. However, they do not allow to encode arbitrary parameters inside the model. For example, we are interested in encoding contextual information like certain task or environment parameters inside a GMM and generalize the demonstrated task with respect to these parameters. When using movement primitives, this can only be achieved by applying a hierarchical approach, e.g., learning an upper-level policy that encodes the DMP hyper-parameters together with contextual information on the task.

WBC is today a well-understood topic and a large variety of approaches exist. However, some open problems persist, see [MS19] for a discussion. One issue is that, although the individual controllers might be stable, the accumulation of control actions resulting from different tasks may result in unstable behavior and a formal proof is difficult. Another open issue is improving the speed of optimization-based approaches when many degrees of freedom and constraints are considered. In this context there is a lack of proper benchmarks in order to compare available solvers. Furthermore, the integration of different trajectories computed by planning modules with reactive controllers for e.g., obstacle avoidance and balancing with

the aim of providing feasible overall trajectories is an open research problem. Apart from the implementation issues of whole-body controllers, most of the developed solutions focus on state estimation and control. They introduce hand-crafted whole-body controllers, which are carefully designed to tackle individual problems, and lack generality. To develop more autonomous robots that operate in dynamic environments and adapt to novel situations, the link to higher-level intelligence like reasoning and machine learning concepts is required. In this thesis, we attempt to provide such a link by introducing methods to automatically derive, adapt and optimize task constraints for WBC.

One of these methods is based on PbD, which has complementary strengths compared to Whole-Body Control. While PbD offers an intuitive way for non-experts to program a robot, WBC allows expert users to specify complex, multi-objective tasks on robots with redundant dof. The combination of both approaches provides an intuitive way to program complex tasks on redundant robots and adapt them according to the current situation. In the following chapter, we provide an overview on the state of the art on PbD-based approaches for automatic derivation of whole-body controllers, as well as other methods for automatic or intuitive specification of tasks for WBC.

Whole-Body Control is a well-established approach to program redundant robots and it has been used to solve complex robotic tasks like balancing [Nor+15], climbing stairs and ladders [Fen+15], jumping [Kui+16] or mobile manipulation [Van+12]. In WBC, complex tasks can be composed from simpler task descriptors, which are posed as constraints to an optimization problem. This separates the task description from the details on its execution and leaves the programmer the job of specifying tasks constraints and define their relative importance by the means of priorities. A lot of expertise is required to select the task models, define priorities, and tune parameters. And even then, the result is usually not satisfying right away, the task parameters must be tuned in a time-consuming trial-and-error process. The manually programmed tasks usually perform well only in a certain context. If the task or environment changes, the task description must be adapted.

The state-of-the-art review presented in this chapter has partly been published [MK21]

In this thesis we want to provide a way to simplify the task specification process in WBC and generalized task descriptions to previously unseen situations. This chapter provides an overview of existing works in this field, which can be roughly classified into the following categories (see also Table 3.1):

- ▶ **Programming by demonstration-based** approaches that derive task constraints for WBC from data acquired in user demonstrations
- ▶ **Optimization- or Reinforcement Learning-based** approaches that optimize task constraints given an initial guess
- ▶ **Knowledge-based methods** that apply symbolic reasoning to infer task constraints for WBC approaches in a specific situation

This chapter is organized as follows: We first review the state of the art for the three categories in the Sections 3.1, 3.2 and 3.3. While we provide a review on all three categories, this thesis focuses on developing methods that fall into the first two. Section 3.4 mentions some further approaches that do not fit into one of the three categories. Apart from that, there is also a relation of the methods developed in this thesis to classical behavior learning, which is discussed in Section 3.5. Finally, we provide a brief discussion and summary in Section 3.6.


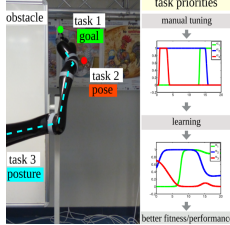
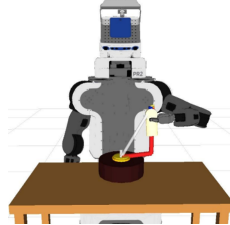
Method	PbD-based	Optimization/RF-based	Knowledge-based
Description	Derive task constraints and associated (strict or soft) task priorities from data obtained in user demonstrations	Derive task priority functions using black-box optimization or reinforcement learning given an initial guess	Use symbolic reasoning to parameterize Whole-Body Controllers
Example	 <p>Efficient Learning of Constraints and Generic Null Space Policies [Arm+17]</p>	 <p>Learning soft task priorities for control of redundant robots [Mod+16b]</p>	 <p>Knowledge-based Specification of Robot Motions [TBB14]</p>
References	[How+09; Arm+17; Arm+18; LHV15; LRH17; CGB07; CB08; CB09; Cal16; RCC14; Ure+15; FBB16; Sil+19; PDA19]	[Cha+18; Spi+17; Mod+16a; Mod+16b; LPS16; LPS15; LPS14; DRS15; DRS16]	[BKB13; KB12; TBB14; Son+10; Lei+16; Lei+14]

Table 3.1: Overview of the state of the art on automatic derivation of task constraints

### 3.1 PbD-Based Approaches

Programming by demonstration, also referred to as imitation learning, is an end user approach to learn robot tasks without explicit programming, namely by demonstrating the task of interest and transferring it to the target system. Demonstrations can be in the form of guiding the robot along the desired trajectory (kinesthetic teaching), teleoperation using exoskeletons or joypads, motion capturing, gesture recognition or other input devices. Apart from providing an intuitive user interface for robot programming, PbD may also reduce the search space for learning, either by starting from an observed (good) example or by discarding from the search space what is known as a bad solution [Bil+08]. PbD has been applied to learn various robotic manipulation skills, especially in combination with reinforcement learning. Examples include teaching a robot to play ball-in-a-cup [KP14] or flipping a pancake [KCC10].

**Classical PbD** Most existing PbD approaches try to learn a single task, usually on a six or seven dof robot arm. In contrast to that, we attempt to learn a whole-body problem including multiple tasks that shall be executed in parallel on a robot with many dof. For example, we might want to jointly learn a manipulation task and obstacle avoidance. Since the learned tasks might be conflicting, we additionally must derive their relative importance from the acquired data. Thus, the problem we consider is twofold: Estimate the task constraints in terms of a

motion description and derive their associated task priorities, i.e., their relative importance, from data. In addition, we want the constraints to generalize over new situations (contexts), so that the user does not have to specify similar tasks repeatedly and the robotic system behaves more autonomously in novel situations. The problem of learning task constraints in WBC from user demonstrations has been tackled before in literature and we give an overview hereinafter.

**Policy Learning with Variable Constraints** Several works consider the problem of learning constrained motion from observations. Howard et al. [How+09] extend classical policy learning to account for variable constraints the demonstrated task is subject to. The authors provide a method that learns the underlying (unconstrained) policy from constrained motion data, which enables them to generalize the policy to novel, previously unseen constraints. They evaluate their approach in a dual-arm reaching scenario with varying obstacles on the humanoid robot ASIMO and in a wiping task with variable surface orientations using a seven dof robot arm. There are a couple of follow-up works [Arm+17; Arm+18; LHV15] which also focus on wiping or polishing tasks and generalize the acquired task constraints to novel, previously unseen contact surfaces with different orientation or curvature. Another work in this area evaluates a similar approach in the context of grasping with a multi-fingered robotic hand [LRH17]. All these works have some similarities with the goals of this thesis: They try to learn from constrained motion and attempt to generalize the learned task constraints over a range of novel situations. However, the presented methods are restricted to a fixed task hierarchy consisting of the task constraints on the highest priority level and the unconstrained policy executed in the associated null space. The number of applications is therefore limited. In contrast to that, we attempt to learn both task constraints and the respective priorities from demonstrations and do not make assumptions about the underlying task hierarchy.

**Task-Parameterized Gaussian Mixture Models** While the aforementioned approaches rely on a weighted combination of linear models like radial basis functions (RBFs) for representing motion, another line of research uses probabilistic models to extract task constraints from user demonstrations. Calinon et al. [CGB07] introduce a PbD framework for learning a task on a humanoid robot. The approach relies on Gaussian Mixture Models (GMMs) for encoding the trajectories and Gaussian Mixture Regression (GMR) for generalizing the resulting behaviors to different contexts. While the GMR algorithm has first been described in [GJ94], the term Gaussian Mixture Regression has first been introduced by Calinon et al. [CGB07]. Compared to the previously described approaches, this method does not assume a fixed, strict task hierarchy, but estimates the relative importance of task constraints from the statistical variance observed in the demonstrations. This importance is encoded in terms of weighting matrices in the inverse kinematics solution. The approach is evaluated in different manipulation tasks on a humanoid robot, namely manipulating a chess piece, a bucket, and a piece of sugar. The system can generalize to a certain extent over different initial positions of the manipulated object. Various extensions of the probabilistic PbD-framework have been proposed. For example, Calinon and Billard [CB09] extend the method to simultaneously treat constraints in joint and task space. The authors aggregate their efforts in a framework for learning and generalizing skills on a robot called Task-Parameterized Gaussian Mixture Models (TP-GMMs). Calinon [Cal16] present an overview on applications and extensions of this framework. While classical TP-GMM only considers motion, the approach presented



by Rozo et al. [RCC14] uses a dynamical system to encode the robot task so that positions and contact forces can be learned simultaneously. Thus, the dynamics of the manipulation task can be accounted for, a fact that is evaluated in a human-robot co-manipulation scenario. Silvério et al. [Sil+19] extend the TP-GMM framework to additionally learn the evolution of strict task priorities from demonstrations. For this purpose, they exploit the variability between the demonstrations in task space when different strict candidate hierarchies are employed. From this variability, they obtain a joint level controller using soft weighting of strict hierarchies. Thereby, hierarchies that result in a large variance for the task constraints contribute less to the solution and vice versa. The approach is demonstrated on a humanoid robotic system, and it is shown that task priorities can be generalized to different target locations. In contrast to the previous methods, we develop an approach that allows us to estimate task constraints together with their associated soft priorities and additionally generalizes them to novel situations. The generalization capabilities of previous methods are mostly limited to varying start and end positions of the demonstrated motions, whereas we want to deal with more complex variations of the demonstrated task. Finally, the existing approaches describe the context by means of real-valued vectors of task parameters, e.g., frame transformations. In contrast to that, our approach also allows categorical context variables, e.g., use right/left arm or allow/forbid object tilting.

**Others** There are other probabilistic PbD-approaches that estimate task constraints from demonstration and that are not based on TP-GMM. Ureche et al. [Ure+15] propose a method to learn hybrid force-position tasks from user demonstrations. They extract the parameters of a hybrid impedance controller (reference frame, variables of interest and stiffness) from user demonstrations of a manipulation task like grating a vegetable. The approach generalizes over different initial robot and object positions. Compared to the approaches investigated in this thesis, the authors do not consider learning of task hierarchies, which is an important aspect in case of conflicting tasks. Perico et al. [PDA19] combine a constraint-based control framework with methods from programming by demonstration. They use Probabilistic Principal Component Analysis (PPCA) to represent a demonstrated trajectory. The demonstrated trajectories are integrated as a constraint into eTaSL [AD14], which is a high-level extension of the iTaSC framework. The variance inherent in the demonstrations is used to estimate the stiffness settings of the robot's joint-level impedance controller in order to guide the human operator to the target position. Other task parameters, like, for example, the task priorities, must be chosen manually by the human expert. In contrast, we use data obtained in the user demonstrations to automatically get an estimate of the task priorities. Furthermore, the approach developed by Perico et al. [PDA19] allows only limited generalization, e.g., a varying target position of the end effector. In contrast, we develop a solution that can generalize over complex task parameters like the properties of the handled objects. Fang et al. [FBB16] use Random Forest Regression (RFR) in combination with WBC to learn pouring water into a container. Training data is generated by demonstrating the task in an interactive simulation environment. In contrast to this approach, which seems designed to solve the pouring task, we attempt to provide a more general framework that works on multiple tasks and situations.



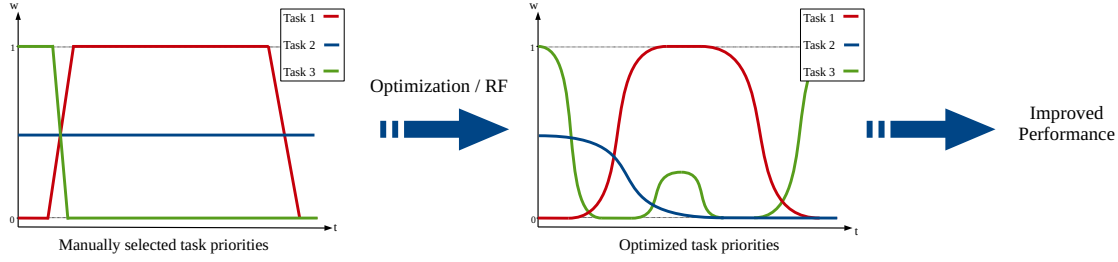


Figure 3.1: Learning task priority functions: Optimized task priorities yield improved WBC performance compared to manually selected ones, drawing inspired by Modugno et al. [Mod+16b].

### 3.2 Optimization- and RF-Based Approaches

Another way to simplify the life of the human expert in WBC is to automatically derive task constraints using black-box optimization or Reinforcement Learning (RF) with the goal of improving performance in terms of stability, compatibility, robustness, or safety. For example, optimization or RF can be used to compute the soft task priorities of a Whole-Body Controller. In this case, the desired trajectories for each task are given, e.g., by a motion planner. Compared to strict priorities, soft task priorities are much harder to tune. On the one hand, the performance may strongly depend on the selected values. For example, the system might become unstable when the chosen values for a certain task are too high or too small. On the other hand, each dimension of a task constraint may have different optimal values of the task priorities. For example, the motion or force along a certain axis might be more important than for other axes. This implies that all soft task priorities must be tuned individually and, depending on the problem, their number might be quite large. Finally, optimal behavior might only be achieved if the task priorities vary over time. For example, when approaching an obstacle, the respective task priorities for collision avoidance should be increased and decreased again if the robot withdraws from the obstacle. Another example is the execution of task sequences. When transitioning from one task to the next, the respective task constraints can be faded-in and -out smoothly. Thus, they must have time-varying task priorities, which are obviously much harder to compute than fixed values. Figure 3.1 shows the general idea of learning time-depending task priority functions.

Different works exist that attempt to learn soft task priorities using stochastic optimization. For example, the approach described by Charbonneau et al. [Cha+18] increases robustness with respect to perturbations and varying conditions when transferring the task priorities to the real robot. The (time-fixed) task weights are learned using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Höt01]. As a fitness function the authors use a combination of task space tracking error, joint torque and Zero Moment Point (ZMP) deviation from the center of the support polygon. The results are evaluated on the humanoid iCub robot [Met+08]. In contrast to that, Modugno et al. [Mod+16b] present an approach to learn time-varying task priority functions, modeled as a mixture of radial basis functions (RBFs). The authors also use CMA-ES with a combination of task space tracking error and joint torque as a fitness function to learn the parameters of the RBFs. Experiments are performed on a seven dof robot arm in a scenario, where the robot must reach different goal positions in the presence of obstacles. In a follow-up work the authors benchmark their solution with

different CMA-ES variants and apply it to the iCub humanoid [Mod+16a]. The objective is here to learn task priority functions that ensure that physical constraints are never violated. CMA-ES is also employed by Dehio et al. [DRS16] to optimize the mixing coefficients in a torque-based multi-objective controller. As a fitness function they use a combination of minimum task space error, execution time and energy consumption.

Another line of research deals with finding optimal task trajectories for multiple competing objectives in WBC using stochastic optimization or Reinforcement Learning. Lober et al. [LPS14] describe each task as a Dynamic Movement Primitive (DMP) [INS02]. The parameters of the DMPs of all tasks are optimized using a black-box optimization algorithm in order to minimize task incompatibilities. The approach is demonstrated on the iCub humanoid. Later, the same authors use RF to optimize the location and temporal evolution of waypoints when manipulating objects with the iCub humanoid [LPS16]. As a cost function they use a combination of trajectory tracking error, the position error in the target location and energy consumption. This way, they can overcome task discrepancies, and increase the probability of a successful task execution.

Compared to our work, the optimization and RF-based approaches as described above focus on deriving either soft task priorities or task constraints in terms of motion descriptions. In contrast to that, we develop an approach that is able to derive task constraints and priorities simultaneously from the same data set. Furthermore, the previous methods provide only limited generalization capabilities as they perform optimization with respect to a particular situation. Our approach on the other hand attempts to generalize task constraints over a variety of situations. Furthermore, all approaches mentioned above are either applied in a very restricted context, only in simulation or for simple robot tasks. Compared to that we are striving for methods that can be applied to more general robot control problems, are able to deal with considerable number of parameters and contexts and can be easily applied on real robots.

### 3.3 Knowledge-Based Approaches

Other approaches use high-level reasoning mechanisms to parameterize Whole-Body controllers. We refer to this class of approaches as knowledge-based methods here. The idea is to use background knowledge to infer WBC task parameters given a certain task and situation. This way, complex task sequences can be planned and executed using WBC approaches. Figure 3.2 shows the general idea of using knowledge-based approaches to parameterize Whole-Body controllers.

In the work presented by Leidner et al. [Lei+14; Lei+16], a reasoning framework is combined with a torque-based WBC approach to solve different compliant manipulation tasks on the humanoid robot Justin [Fuc+09], namely wiping a window, scrubbing a mug, and sweeping the floor. Task parameters like e.g., contact stiffness, the control hierarchy and force limits are stored together with the model of the manipulated object and inferred using a combination of geometric and symbolic reasoning. The approach provides abstract, object-centric action templates for complex manipulation tasks, which can be transferred between different robots. Generalization capabilities are provided in the sense that the task trajectories are defined in object space, such that the location of the manipulated object can be varied without changing

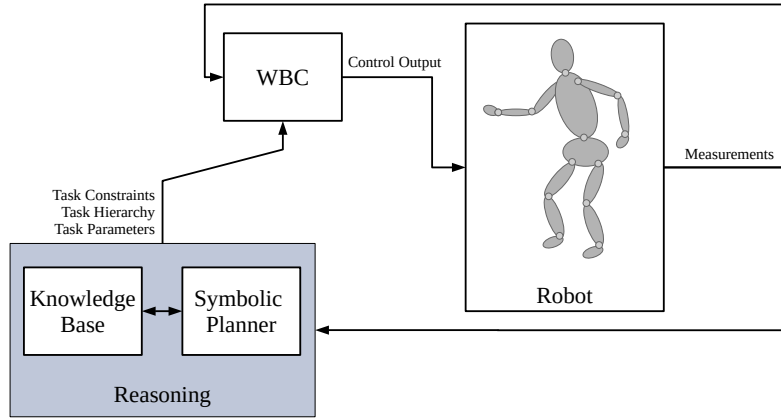


Figure 3.2: Knowledge-based approaches parameterize Whole-Body Controllers using symbolic reasoning, drawing inspired by Leidner et al. [Lei+16]

the task specification. However, the concrete task trajectories, task hierarchies, stiffness parameters and other task parameters still must be selected by hand. At least an allowable range of values must be given by the human expert. Thus, the presented approach does not automatize the process of task specification, it only shifts the problem of parameter selection to a more user-friendly level. Other knowledge-based methods have similar shortcomings. For example, the approach presented by Tenorth et al. [TBB14] the reasoning framework CRAM (Cognitive Robot Abstract Machine) [BMT10] is used in combination with abstract constraint descriptions and a WBC approach to specify household tasks like pouring pancake mix into a frying pan on the PR2 robot [Boh+11]. As the authors argue, the motion descriptions provided still contain hand-crafted numbers, which limits the generalization capabilities of the approach. Kresse and Beetz [KB12] and Bartels et al. [BKB13] both present a system that attempts to bridge the gap between symbolic action representation and a motion control approach based on task functions. Constraints are defined using geometric features, providing a symbolic, constraint-based movement description language. The approach is evaluated in a pancake making scenario. Again, as with the previous approaches, an automatized selection of numerical values given a symbolic action description is not considered.

Another interesting approach is introduced by Song et al. [Son+10]. The authors use a Bayesian network to model, in a probabilistic manner, task, object, action and constraint features as a joint probability distribution. While the task is described by symbolic variables like e.g., "hand-over" or "pour", the object, action and constraint features are described by numeric values like object size, grasp configuration or grasping constraints. The Bayesian network is trained using data generated by a grasp planner and a human expert that labels the suggested grasp configurations of the planner. The initialized network can be used for decision making. For example, given the perceived object and a certain task, the associated action features, and task constraints like e.g., the optimal grasp configuration and object enclosure can be inferred. While sharing some similarities with the goals of this thesis, namely the possibility to derive task constraints from user demonstrations, the application domain is quite different. The system is only applied to multi-fingered grasping and does not have a relation to WBC.

### 3.4 Other Approaches

There are some other approaches that supply user-friendly interfaces to WBC-based task specification and cannot be classified into one of the three categories. For example, Aertbeliën and De Schutter [AD14] present a symbolic programming language called eTaSL (Expressiongraph-based Task Specification Language). The idea is to provide a flexible and user-friendly way to define and execute robot tasks based on expression graphs. Somani et al. [Som+15] presents a task specification framework based on geometric constraints expressed in CAD models. The framework shall ease task-based robot programming for non-experts in industrial settings. Both approaches may simplify the process of creating WBC-based task descriptions, but they do not automatize it. Furthermore, they do not provide general solutions by any means, but rely on the fact that new tasks can be quickly (re-)programmed by the user.

### 3.5 Relation to Behavior Learning

Behavior learning in robotics is the process of learning robot skills that can generalize over task parameters. As with most machine learning approaches, robot behavior learning is particularly popular in applications where modeling the problem is difficult or even impossible. In behavior learning, a commonly used method is to retrieve an initial trajectory by the means of imitation learning and refine them using RF. Behaviors can be described using a parametric motion model, for example Dynamic Movement Primitives (DMPs) [KP10]. DMPs have been designed to generalize over some meta-parameters like its initial position or duration of motion [INS02]. Adaptation to more complex task parameters can be achieved using hierarchical approaches. Hierarchical approaches apply an upper-level policy that is supposed to generalize over the meta-parameters of the lower-level policy [FM14; Kup+17; WLP17]. Fabisch et al. [Fab+19] provide an overview over the state of the art in behavior learning for robotics, with the focus on real-world applications.

Most of the existing behavior learning methods focus on algorithmic development. The applications are commonly limited to learning a single task, which is executed on a robotic arm with six or seven dof. In contrast to that, this thesis focuses on learning task constraints for multi-task scenarios on more complex systems with kinematic tree structure.

### 3.6 Discussion and Summary

In the earlier sections, we gave an overview on the state of the art in learning task constraints for Whole-Body Control of robotic systems. Learning task constraints for WBC instead of manually specifying them can have different objectives. On the one hand, learning task constraints can simplify or fully automatize the task specification process in WBC and thus reduce the burden of the human programmer. On the other hand, it can generalize task descriptions to novel situations and automatically adapt the robot behavior without the need to re-program the system repeatedly. As a result, the target robot will be more adaptive and autonomous, especially in dynamically changing environments.

Approach	Use PbD	Task Constraints	Task Priorities	Soft/Strict Priorities	Generalization
[Mod+16b],[Mod+16a]	-	-	✓	soft	-
[LPS16]	-	✓	-	soft	-
[DRS15]	-	-	✓	soft	-
[Arm+17]	✓	✓	-	strict	✓
[PDA19]	✓	✓	-	soft	✓
[Sil+19]	✓	✓	✓	strict	✓
[FBB16]	✓	✓	-	soft	✓
Our approach	✓	✓	✓	soft/strict	✓

Table 3.2: Comparison of the main approach for learning task constraints followed in this thesis with the state of the art.

We classified the existing state of the art methods in PbD-based, optimization- / RF-based and knowledge-based approaches. PbD-based approaches derive task constraints and/or the associated task priorities from data obtained in user demonstrations. In contrast, optimization- and RF-based approaches derive either task priority functions or task trajectories given an initial guess. Finally, knowledge-based methods use symbolic reasoning to parameterize Whole-Body Controllers.

Table 3.2 compares the main method developed in this thesis with the state of the art according to the type of learning (PbD vs. other e.g., RF), the learning target (task constraints, task priorities or both), whether the approach applies to strict or soft priorities and whether the approach achieves generalization capabilities. According to the table the most similar approach to ours has been presented by Silv rio et al. [Sil+19]. However, this approach demands that the user selects candidate hierarchies in terms of projection operators in advance. Compared to that, our approach only requires the selection of task relevant coordinate frames, which is trivial in every case. Furthermore, the approach relies on strict task hierarchies, which does not work well on over-constrained problems. In summary, the main differences and innovations of the approaches developed in this thesis compared to existing state of the art methods are

- In contrast to classical PbD applications and behavior learning approaches, which often focus on a single task executed on a manipulator arm, we attempt to learn multiple parallel tasks applied on complex robots with kinematic tree-structure.
- Compared to most existing methods that combine machine learning and WBC, we attempt to learn both, task constraints and associated priorities at the same time.
- We attempt to generalize both, task constraints and priorities with respect to previously unseen contexts, where the context description may be real-valued or categorical. In contrast, existing methods only allow minor contextual variations, or they focus on simple contexts like start and end positions of the demonstrated motion.

The approach for learning task constraints, which based on PbD, is presented in the following chapter.



# Learning Context-Adaptive Task Constraints from Demonstration

# 4

In this chapter we introduce a programming by demonstration approach for whole-body controllers. The approach automatically derives a part of the optimization problem, namely task constraints and the associated task priorities, from data acquired in user demonstrations. The demonstrations are performed in varying conditions, which we refer to as contexts. From the acquired data, we derive probabilistic models that generalize task constraints and priorities over various contexts and adapt robot behavior to novel, previously unseen situations. We evaluate our approach by learning several manipulation tasks on an industrial dual-arm robot and on a humanoid system.

Parts of this chapter have already been published [MK21]

The original contributions in this chapter are:

- i. A PbD approach to derive task constraints and the associated soft task priorities for a whole-body controller from user demonstrations.
- ii. An adaptation method based on Gaussian Mixture Regression (GMR) that generalizes the acquired task constraints and soft priorities with respect to context changes.
- iii. An alternative to the GMR approach based on Probabilistic Movement Primitives.

This chapter is organized as follows: Section 4.1 explains the motivation for developing the approaches. Section 4.2 gives an overview on the PbD workflow. In Section 4.3, we describe the WBC implementation used and the modelling of task constraints. Section 4.4 illustrates the process of data acquisition and processing. In Section 4.5, we describe a method for estimating task constraints with soft priorities and present results using an industrial dual-arm robot. In Section 4.6 we compare the proposed approach to a method based on Probabilistic Movement Primitives (ProMPs). We evaluate the approach on a humanoid robot. Finally, Section 4.7 provides a brief discussion and outlook.

## 4.1 Motivation

Whole-Body Control is a flexible and well-adopted framework to specify and control complex tasks on redundant robots. Although WBC has been introduced in the context of humanoid robotics, it is meaningful to apply it to any problem, where multiple robot behaviors must be integrated, while considering physical constraints related to the environment, the robot, or the task itself. Especially in robotic manipulation, many tasks can be specified as a combination of simpler subtasks. For example, the task of cleaning a table can be separated into two subtasks: "maintain contact with the table surface" and "follow the wiping trajectory". Now, maintaining surface contact is obviously more important than following an accurate trajectory parallel to this surface. This preference can be formalized by means of task priorities. Assigning a low priority to the trajectory following task allows to perform additional tasks utilizing the remaining robot dof, for example, avoiding collisions with objects on the table surface. Secondary objectives like minimizing energy consumption can be integrated on the lowest priority level.



As already elaborated in previous sections, the design of task models and constraints, as well as the selection of appropriate task priorities is typically done by a human expert in a manual fashion. This process requires knowledge regarding WBC in general, the specific WBC implementation and the problem domain. Even with that background knowledge, the specification is typically a trial-and-error process, which is tedious and prone to errors. Furthermore, the resulting task models are often specific to a certain situation. If the task specification or the environment conditions change, these handcrafted solutions will fail. Especially selecting incorrect task priorities may lead to a bad control performance, as shown in Section 1.1, or even to completely different robot behavior. An automated procedure to derive task constraints and the respective task priorities for WBC may strongly reduce the work for the human programmer and lead to better results. The latter is especially true if the developed approach is context-adaptive, i.e., if it can automatically adapt the whole-body controller to context changes. Context changes refer to the variations that a robotic task may be subjected to and can be classified into three different categories:

1. Variations of the task itself, e.g., the goal position of the end effector, orientation constraints when carrying an object, etc.
2. Changes in the environment of the robot, e.g., the dimension or the shape of a manipulated object, the presence of obstacles, etc.
3. Changes regarding the robot in use, e.g., different robot morphologies (single arm vs. dual arm, stationary vs. mobile, number of dof, etc. ...), defective actuators/sensors, etc.

In the following sections we introduce a programming by demonstration (PbD) approach to automatically derive task descriptions for Whole-Body Control. Programming by demonstration and Whole-Body Control (or model-based control in general) have complementary characteristics [PDA19]. PbD provides the user an intuitive interface to specify novel tasks on a robotic system without requiring any expertise in robotics or programming skills. In contrast, WBC allows a human expert to program reusable, sensor-based skills for complex robots in dynamic environments. The combination of both enables non-experts to create solutions for complex and adaptive behavior on redundant robots, while preserving the positive features of stand-alone whole-body controllers.

## 4.2 Approach

We first supply an overview of our approach and the PbD workflow. The goal of the presented method is to reduce the effort for the human programmer to design the optimization problem of a whole-body controller. An important part of the optimization problem is the task constraints and their respective task priorities. In general, many different constraints act on a robotic system when performing a manipulation task. These can be related to the robot (physical constraints like joint limits, self-collision, ...), to the environment (stationary or moving obstacles, contact surfaces, manipulated objects, ...) and the task itself (goal positions, objects that may not be tilted, ...). The idea of the presented approach is that the demonstrated task is subject to constraints which are consistent over multiple demonstrations. Thus, they can be extracted by identifying the invariant features in the acquired data. As an example, consider again the task of wiping a table surface. The surface constrains the motion of the robot, which is reflected in the data acquired in the user demonstrations. Even if we perform different wiping motions, the motion vertical to the table surface will have a low variance in all



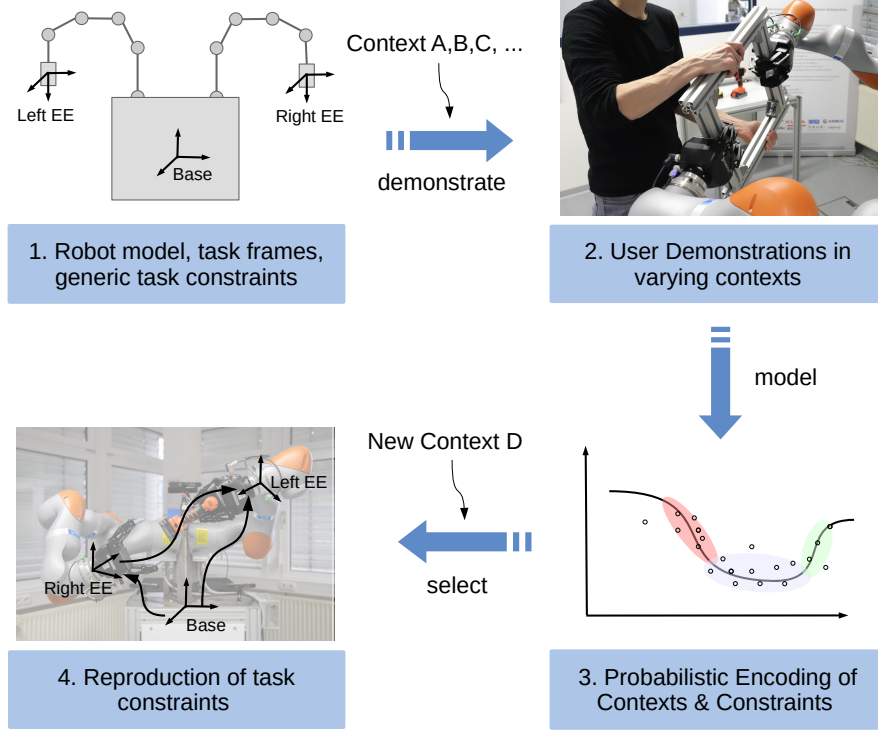


Figure 4.1: Overview of the approach on learning context-adaptive task constraints from user demonstrations

demonstrations. This variance can be mapped to a task constraint with high priority in a WBC framework, while other tasks are assigned lower priorities. This way, the relevant aspects of the task are preferred by the whole-body controller and other tasks are subordinated, leaving redundant dof free for other important tasks. To generalize the task constraints with respect to previously unseen situation, we perform the user demonstration in different contexts. From the acquired data, we can derive probabilistic models that generalize the extracted task constraints over context changes like the height of the table, the wiping direction, or the tilt angle of the contact surface. Thus, the whole-body controller learns to adapt to novel, previously unmet situations.

Figure 4.1 shows an overview of the proposed PbD approach, which consists of the following steps:

1. The kinematic model of the robot is assumed to be known. The human expert must select in advance only a couple of relevant coordinate frames, referred to as task frames here. (e.g., the base of the robot, the gripper frames, or an object-related coordinate frame). The selected task frames are used to model the task constraints, i.e., each task constraint describes the relative motion of two task frames (see Section 4.3).
2. We perform several user demonstrations in the form of kinesthetic teaching in different contexts (see Section 4.4). The context may vary over time. However, here we assume that the context is time-invariant, at least within a single demonstration. In each demonstration we record the available task constraints.
3. We model the joint probability distribution over context variables and task constraints as a Gaussian Mixture Model (GMM) (see Section 4.5).

4. From this joint distribution, we can generalize the task constraints, as well as their associated task priorities and reproduce them in a novel, previously unknown context using Gaussian Mixture Regression (GMR) (see Section 4.5).

### 4.3 WBC Framework and Task Constraints

The WBC approach we use here is a variant of the method presented by Smits et al. [Smi+08]. As in their approach, we describe robot tasks by imposing constraints on the relative twist between two rigid bodies. Each body is thereby described by a unique coordinate system, denoted as task frame [BBD03].

**Definition 4.3.1** *A task frame is a uniquely defined, task relevant coordinate system relative to which a robot task can be specified.*

Zero twist between two task frames means that the associated bodies are not supposed to change their relative position and orientation during the task. Multiple constraints can be integrated by solving the following instantaneous optimization problem:

$$\begin{aligned} & \underset{\dot{\mathbf{q}}}{\text{minimize}} && \|\dot{\mathbf{q}}\|_2 \\ & \text{subject to} && \begin{pmatrix} \mathbf{J}_{W,1} \\ \vdots \\ \mathbf{J}_{W,P} \end{pmatrix} \dot{\mathbf{q}} = \begin{pmatrix} \mathbf{v}_{d,1} \\ \vdots \\ \mathbf{v}_{d,P} \end{pmatrix} \end{aligned} \quad (4.1)$$

Here  $\dot{\mathbf{q}} \in \mathbb{R}^N$  are the joint velocities of the robot, where  $N$  is the number of joints. The number of task constraints is denoted as  $P$ ,  $\mathbf{J}_{W,i} = \mathbf{W}_i \mathbf{J}_i \in \mathbb{R}^{6 \times N}$  is the weighted task Jacobian of the  $i$ -th task and  $\mathbf{W} \in \mathbb{R}^{6 \times 6}$  refers to a diagonal matrix, which contains the task weights  $\mathbf{w} = (w_1, \dots, w_6)^T$  on its main diagonal. The task weights are also referred to as soft task priorities. The solution of (4.1) is the robot joint velocity that complies with all task constraints of type  $\mathbf{J}_i \dot{\mathbf{q}} = \mathbf{v}_i$  in the best possible manner. This means, the solution ensures that the  $P$  pairs of user-defined task frames move with the desired relative twists  $(\mathbf{v}_{d,1}, \dots, \mathbf{v}_{d,P})^T$  if possible. Like with any other WBC approach, (4.1) is not a global optimization problem but provides the optimal solution at a specific time instant. In every control cycle, the optimization problem is updated with the new  $\mathbf{J}_{W,i}$ ,  $\mathbf{v}_{d,i}$  and solved for  $\dot{\mathbf{q}}$ , which is directly applied to the robot actuators. To impose constraints on the relative poses of two task frames, a set of closed-loop position controllers can be designed around the optimization problem in (4.1). These controllers have the following form:

$$\mathbf{v}_d = \mathbf{v}_r + \mathbf{K}_p \begin{pmatrix} \mathbf{p}_r - \mathbf{p} \\ \theta \hat{\boldsymbol{\omega}}_r \end{pmatrix}, \quad \mathbf{v}_d \leq \bar{\mathbf{v}} \quad (4.2)$$

where  $\mathbf{v}_r \in \mathbb{R}^6$  is the reference (feed forward) twist,  $\mathbf{p}_r, \mathbf{p} \in \mathbb{R}^3$  the reference and actual position,  $\theta \hat{\boldsymbol{\omega}}_r \in \mathbb{R}^3$  the rotational difference between reference and actual pose,  $\bar{\mathbf{v}}$  the controller saturation and  $\mathbf{K}_p \in \mathbb{R}^{6 \times 6}$  a diagonal matrix containing 6 feedback gain constants. The control output  $\mathbf{v}_d$  is fed into (4.1) for each task constraint. Assuming that the kinematic model of the robot is given, the representation in (4.1), (4.2) includes the relative pose and

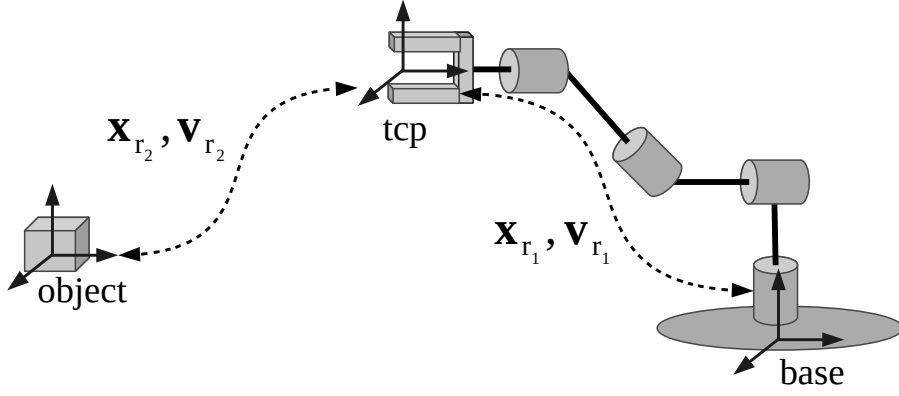


Figure 4.2: Modeling of task constraints.

twist of two task frames, as well as the soft task priorities as main design parameters. We refer to the tuple of relative pose and twist of two task frames as *task constraint* here.

**Definition 4.3.2** A task constraint is described by the relative pose and twist of two task frames  $(\mathbf{x}(t), \mathbf{v}(t))$  and its associated (soft) task priorities  $\mathbf{w}(t)$ .

The task constraints and their associated priorities are functions of time, so they may change during task execution and can be described as time-indexed trajectories. Figure 4.2 illustrates the concept of task constraints that we employ here. Each task constraint corresponds to a virtual kinematic chain connecting the corresponding task frames. The controller in (4.2) regulates the relative pose and twist of two task frames, while the soft task priorities employed in (4.1) control the relative importance of each task variable within the solution. As an example, consider the case where robot orientation is not relevant to us. Here we can set the corresponding soft task priorities to zero and only the position of the robot will be considered in the solution. Using soft task priorities here means that the tasks are not hierarchically organized as in the Whole-Body Control framework introduced by Sentis and Khatib [SK06], but the solution will be a weighted combination of the different tasks. If we consider the over-constrained case, we have more task variables than available robot dof. Here, the solution approximates the demanded tasks. The accuracy of each task depends on the values of the corresponding soft task priorities. The higher these values, the more accurate a task is fulfilled. The solution of (4.1) is computed using a standard QP solver [FBD08]. Throughout the entire chapter, we assume that a sufficiently accurate kinematic model of the robot is known, providing the task Jacobians required in (4.1). Such a model can be obtained from CAD export and refined through kinematic calibration [HW96]. We also assume that the robot model contains only rigid bodies, where each rigid body is assigned a unique coordinate frame.

The WBC approach used here can model several types of task constraints like joint limits, collision avoidance or contact forces. However, for the sake of simplicity we consider only task constraints as described in Definition 4.3.2. In the following sections we will illustrate an approach to automatically derive the quantities  $(\mathbf{x}(t), \mathbf{v}(t))$ , as well as  $\mathbf{w}(t)$  from user demonstrations and adapt them to novel, previously unseen contexts.

## 4.4 User Demonstrations

For data acquisition, we use kinesthetic teaching. As already mentioned in Section 2.2.1, the advantage of kinesthetic teaching compared to other data acquisition methods like human motion tracking is that we do not have to bother about workspace transformations and reachability issues afterwards. In each demonstration we record the current context and the relative poses and twists between all pairs of task frames. For now, we refer to these pose/twist pairs as *candidate constraints* ( $\mathbf{v}^*(t), \mathbf{x}^*(t)$ ) in order to distinguish them from the actual and the estimated task constraints. The importance of a candidate constraint for the demonstrated task will be determined by its associated priority  $\mathbf{w}(t)$ . Both the task constraints and the priorities shall be estimated from the acquired data.

In the following we describe the representation of the data acquired in user demonstrations and the necessary preprocessing steps.

### 4.4.1 Representation of Context

Contextual information can be used to characterize the state of the world and aid in robotic decision making. According to [Tur98], contextual information can be classified in three categories: environmental information, task-related information, and agent self-knowledge. Another critical point is how to represent contextual knowledge. Bloisi et al. [Blo+16] distinguish three different classes of context representation structures. First, there is embedded context representation, which describes features on a subsymbolic level. Such a representation is typically applied in perceptive systems. Secondly, there are logic-based representations, which use declarative knowledge representation languages that allow inference on a symbolic level. Third, probabilistic representations allow reasoning about uncertainty and modeling the degree of belief when selecting a behavior for a certain situation. Most existing approaches for context representation cannot be categorized exactly into one of the three categories but comprise a combination of them.

In our approach, we describe context as a real-valued vector  $\boldsymbol{\kappa} \in \mathbb{R}^C$ , where  $C$  is the number of context variables. Since we model context and task constraints as a joint probability function later, the approach falls into the category of probabilistic context representations. In general, the context may change over time. However, here we assume that the context is time-invariant during a demonstration. As the topic of state estimation and context identification is out of the scope of this thesis, the user currently must specify the context variables for each demonstration by hand. The context variables can be real-valued numbers, e.g., the dimensions of an object, or categories, e.g., single, or dual-arm task execution, allow or forbid tilting the object during task execution, etc. In case of a categorical representation, one-hot encoding is used to model the categories and integrate them into the real-valued vector  $\boldsymbol{\kappa}$ . Table 4.1 shows an example of context variables, which describe the width of the manipulated object, whether the object may be tilted or not and whether it should be manipulated with the right or left arm. The context vectors for these two cases are  $\boldsymbol{\kappa}_1 = (0.3, 1, 0, 1)^T$  and  $\boldsymbol{\kappa}_2 = (0.5, 0, 1, 0)^T$ .

Other context-adaptive approaches mostly allow an adaptation with respect to varying start- and end positions of learned motions. In contrast to that we focus on major context changes that may lead to a completely different characteristic of the demonstrated tasks. Considering

Context #	Object width	Allow Tilt	Right Arm	Left arm
1	0.3m	1	0	1
2	0.5m	0	1	0

Table 4.1: Example context variables

again the task of wiping a table surface, a context change might be related to the type of wiping motion, which can be executed as a circular or a linear motion.

#### 4.4.2 Representation of Task Constraints

During user demonstrations we acquire candidate constraints  $(\mathbf{v}^*(t), \mathbf{x}^*(t))$  for each pair of selected task frames, which have the form time-indexed trajectories. Afterwards these trajectories are resampled and temporally aligned. Furthermore, the time variable is normalized to  $[0, 1]$ .

As a next step, we convert the rotational part of the pose trajectories  $\mathbf{x}^*(t)$  into a representation which is suitable for regression. Many different forms of describing orientations can be used in robotics, each of which has certain advantages and disadvantages. Euler angles are a minimal representation composed from three consecutive elemental rotations. Though being minimal, this form of representing orientation is not unique, i.e., different sets of Euler angles may represent the same orientation. Also, Euler angles suffer from singularities and, thus, have a discontinuous representation space. This is particularly problematic for regression as the mean of two sets of Euler angles is not necessarily the mean of the represented rotation. In contrast, orthogonal rotation matrices have a continuous representation space. However, they are inherently over-parameterized and orthonormality must be enforced during regression, e.g., by applying Gram-Schmidt orthonormalization [Mat21]. A compromise can be obtained by using unit Quaternions, which are more compact than rotation matrices and numerically stable. However, they are also not unique, i.e., there are two different quaternions that represent the same orientation. Furthermore, they also have a discontinuous representation space and unit-length must be ensured during regression. Instead, we choose a representation based on elements of the Lie algebra  $so(3)$ , which is a 3-dimensional representation of rotations. The  $so(3)$  is tangential to  $SO(3)$ , the space of  $3 \times 3$  orthogonal rotation matrices. Any rotation matrix can be projected to  $so(3)$  using the logarithmic map [LP17]:

$$\log(\mathbf{R}) = [\hat{\omega}] \theta \quad (4.3)$$

with

$$\theta = \cos^{-1}\left(\frac{1}{2}(\text{tr}(\mathbf{R}) - 1)\right), \quad \theta \in [0, \pi] \quad (4.4)$$

$$[\hat{\omega}] = \frac{1}{2 \sin(\theta)}(\mathbf{R} - \mathbf{R}^T), \quad |\text{tr}(\mathbf{R})| \neq 1 \quad (4.5)$$

Here,  $[\hat{\omega}]$  denotes the skew-symmetric matrix form of the unit rotation axis  $\hat{\omega}$  and  $\theta$  is the rotation angle. The resulting rotation vector  $\hat{\omega}\theta$  provides a three-dimensional representation of the rotation described by  $\mathbf{R}$ . According to Hartley et al. [Har+13], this representation will be unique if we restrict  $\theta \in [0, \pi]$ . However, two particular cases must be considered,  $\theta = 0$  and  $\theta = \pi$ . In each of these cases, the rotation axis inverts its sign. To deal with these situations, we first make sure that the rotational part of the pose trajectory starts in the upper half of  $SO(3)$  by setting  $\omega_z \geq 0$ . Then we iterate each point in the rotational part of the pose trajectory and apply  $\hat{\omega}^* = -\hat{\omega}$ , as well as  $\theta^* = (2\pi - \theta)$  for all the remaining points whenever  $\hat{\omega}$  changes its sign. The resulting three-dimensional orientation trajectory will be continuous. An advantage of using  $so(3)$  elements to represent rotations is that averaging is possible just as it is for elements of  $\mathbb{R}^3$  if the boundary cases  $\theta = 0$  and  $\theta = \pi$  are handled separately. Another reason for choosing this representation is that we want to estimate soft task priorities from the variance in the user demonstrations. The soft task priorities in (4.1) are six-dimensional, where three entries correspond to the linear and three entries to the angular part, respectively. Thus, it is beneficial to represent the rotational part of the pose trajectories  $\mathbf{x}^*(t)$  in three dimensions as well.

In summary, we represent the demonstrated motion in terms of 6D-candidate constraints  $(\mathbf{x}^*(t), \mathbf{v}^*(t))$ . Each of these pairs describes the relative pose/twist between two task frames and can be used as input to the controller in (4.2). The controller output represents a constraint in the optimization problem (4.1) with 6 constraint variables, respectively. The dataset obtained from  $D$  user demonstration  $\xi = \{\mathbf{K}, \mathbf{X}, \mathbf{V}\}$  contains the context data  $\mathbf{K}(t) \in \mathbb{R}^{D \times S \times C}$  and the candidate task constraints  $\mathbf{X}(t), \mathbf{V}(t) \in \mathbb{R}^{D \times S \times M}$ , where  $t \in [0, 1]$  is the normalized time variable,  $S$  the number of samples per demonstration,  $C$  the number of context variables and  $M$  the number of task constraints. The number of task constraints depends on the number of task frames  $F$  as follows:

$$M = \frac{3F!}{(F-2)!} \quad (4.6)$$

For example, for  $F = 3$ , we have 18 constraint variables. As we can see,  $M$  grows strongly with increasing  $F$ . Thus, the problem quickly becomes intractable for large  $F$ . Consequently, the task frames must be selected with care by the human expert.

## 4.5 Estimation of Task Constraints with Soft Priorities

In this section, we describe a method to estimate the task constraints  $(\mathbf{x}(t), \mathbf{v}(t))$  and their respective soft priorities  $\mathbf{w}(t)$  that best reproduce the demonstrated task in a given context  $\kappa$ , which might be unknown to the system. During user demonstrations, we record the task constraints for each pair of task frames and store them in a database. From a training data set  $\xi = [\mathbf{K}, \mathbf{X}, \mathbf{V}]$  we learn a probabilistic model as described in the previous section. Given a certain context  $\kappa$ , the model estimates the task constraints  $(\hat{\mathbf{x}}(t), \hat{\mathbf{v}}(t))$  and associated soft priorities  $\hat{\mathbf{w}}(t)$  that are required to fulfill a certain task. Both, constraints, and priorities, are fed into the WBC framework, which computes the joint velocities that comply with these constraints and sends them to the robot joints. Figure 4.3 shows an overview of the framework including the learning module. Thereby each of the illustrated controllers is an implementation of (4.2), while the solver implements (4.1).

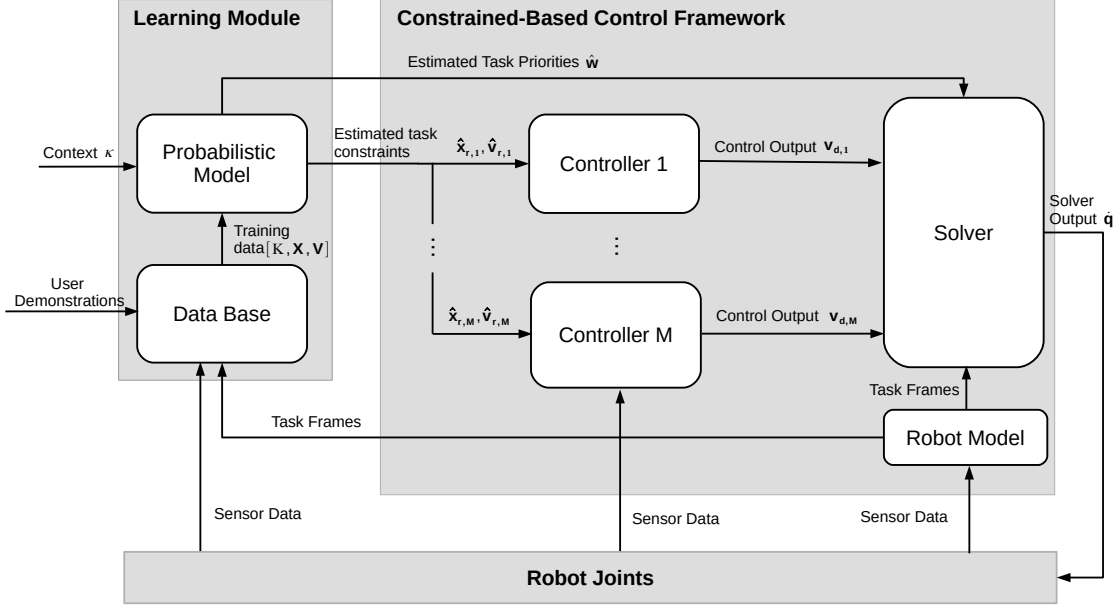


Figure 4.3: Overview of the control framework including learning of context-adaptive task constraints

As we perform  $D$  user demonstrations for each context, each point in the dataset  $\xi(t)$  has a variance  $\sigma_\kappa(t)$ , which describes the variation within the different demonstrations of the same task. We use this variance to get an initial estimate of the soft task priorities. The basic idea is here, that a high variance in the user demonstrations is mapped to low priorities of the corresponding task constraints and vice versa. Figuratively, this means that if certain moving directions have a low variance over all user demonstrations, they are "constrained", either by the task itself, the capabilities of the robot or the configuration of the environment. Thus, it is likely that these aspects are more important for the demonstrated task than those which have a high variability throughout the demonstrations.

#### 4.5.1 Probabilistic Encoding of Context and Constraints

We model the joint distribution of context variables  $\kappa$  and task constraints  $(\mathbf{x}(t), \mathbf{v}(t))$  using Gaussian Mixture Models:

$$p(\mathbf{v}, \mathbf{x}, \kappa | \theta_1, \dots, \theta_K) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{v}, \mathbf{x}, \kappa | \mu_k, \Sigma_k) \quad (4.7)$$

Note that we omit the dependency on the time variable for the sake of readability here. We employ a variant of GMM called Dirichlet Process Gaussian Mixture Model (DPGMM). The model parameters of the DPGMM are described as  $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ , where  $K$  is the number of mixture components,  $\pi_k$  are the mixing weights,  $\mu_k$  the means and  $\Sigma_k$  the covariance matrices of the Gaussian distributions. These parameters are obtained using variational inference. In contrast to a normal GMM, the DPGMM models the mixing weights  $\pi_k$  as a Dirichlet Process. This way, the effective number of mixture components can be inferred from the given data. In practice only a maximum number of mixtures must be



**Algorithm 1** Reproduction of Task Constraints and Soft Task Priorities

---

```

1: Given: Joint distribution  $p(\mathbf{v}, \mathbf{x}, \kappa)$ , Context  $\kappa$ 
2: Start at  $t = 0$ , start pose  $\mathbf{x}_t = \mathbf{x}_0$ 
3: while  $\|\mathbf{x}_e - \mathbf{x}_t\|_2 > \delta$  do
4:   1. Estimate twist:
5:     From  $p(\mathbf{v}|\mathbf{x}_t, \kappa)$  estimate  $\hat{\mathbf{v}}_t$  using GMR
6:   2. Estimate pose
7:     Integrate once to get the corresponding pose estimate as in (4.8)
8:   3. Estimate task priorities:
9:     Compute  $p(\mathbf{x}|\mathbf{v}_t, \kappa)$  using the estimated twist  $\mathbf{v}_t$ 
10:    Compute task priorities using (4.9) and (4.10)
11:   4. Update
12:     Set  $\mathbf{x}_t = \hat{\mathbf{x}}_{t+1}$ 
13: end while

```

---

selected (upper bound) and the algorithm will set the mixture weights of irrelevant Gaussians to near zero. On the downside, the time for training is larger compared to, e.g., EM.

### 4.5.2 Reproduction of Task Constraints

Given the joint probability distribution  $p(\mathbf{v}, \mathbf{x}, \kappa)$ , reproduction of the task constraints and the respective soft task priorities is performed in an iterative manner: Starting from an initial pose<sup>1</sup>  $\mathbf{x}_t = \mathbf{x}_0$ , we estimate the twist  $\hat{\mathbf{v}}_t$  from the conditional distribution  $p(\mathbf{v}|\mathbf{x}_t, \kappa)$  using Gaussian Mixture Regression (GMR) [CGB07] and integrate once to get the corresponding pose estimate:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{x}_t + \hat{\mathbf{v}}_t \cdot \Delta t \quad (4.8)$$

where  $\Delta t$  is the sample time in seconds. Given the estimated twist, we can compute the conditional distribution  $p(\mathbf{x}|\mathbf{v}_t, \kappa)$ . From this distribution with parameters  $\{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$  we can get an estimate of the pose variance for each constraint variable by collapsing the multi-variate Gaussian distribution to a single Gaussian as follows [Tou21]:

$$\mu = \sum_{i=1}^K \pi_i \mu_i, \quad \Sigma = \sum_{i=1}^K \pi_i (\Sigma_i + \mu_i \mu_i^T + \mu \mu^T) \quad (4.9)$$

Note that we omit the time index for the sake of readability here again. From the covariance matrix  $\Sigma$ , we compute the task priorities  $\mathbf{w} = (w_1, \dots, w_M)^T$  as follows:

$$w_j = 1 - \left( \frac{\sigma_j^2}{\bar{\sigma}^2} \right), \quad \forall j \quad (4.10)$$

where  $\sigma_j^2$  are the diagonal entries of  $\Sigma$  and  $\bar{\sigma}^2$  is the maximum variance over all constraint variables. Finally, we set  $\mathbf{x}_t = \hat{\mathbf{x}}_{t+1}$ , estimate the next twist and so on. This process is repeated until convergence to the target pose.

---

<sup>1</sup> Note that  $\mathbf{x}_0$  contains the relative poses of all pairs of task frames, stacked vertically. E.g., if the number of task frames is 3, the dimension of  $\mathbf{x}_0$  is 18.



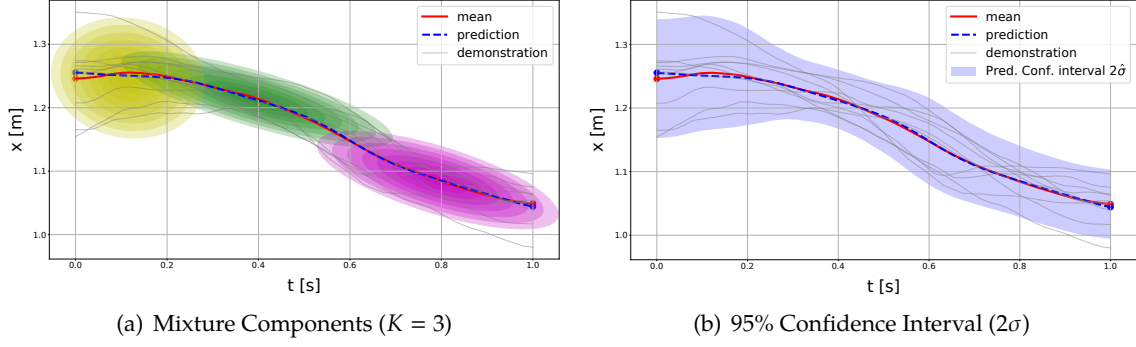


Figure 4.4: Example: Estimated task constraints (x-position) and confidence interval used for predicting soft task priorities.

The procedure for estimating task constraints from GMM is summarized in Algorithm 1. As an input the algorithm requires the context vector  $\kappa \in \mathbb{R}^C$  and the initial poses for all pairs of task frames  $\mathbf{x}_0$ . The algorithm produces the estimated task constraints ( $\hat{\mathbf{x}}(t)$ ,  $\hat{\mathbf{v}}(t)$ ) as output, along with the soft task priorities  $\mathbf{w}(t)$ . As we are using twist commands as variables, the resulting trajectories can be adapted with respect to different starting points.

Figure 4.4 illustrates the reproduction of a motion (only x-position) for a single context. In Figure 4.4(a) the mean and extend of  $K = 3$  mixture components is shown. These mixtures are fitted to  $D = 10$  user demonstrations. The figure also illustrates the predicted trajectory using GMR and the mean of the user demonstrations. Figure 4.4(b) shows the resulting confidence interval  $2\sigma$  computed from (4.9), which is used to estimate the soft task priorities according to (4.10).

### Why GMR?

GMR has several positive properties for motion synthesis on robotic systems. First, in contrast to other regression methods like Gaussian Process Regression (GPR) [Ras04] or Locally Weighted Projection Regression (LWPR) [VDS05], GMR does not model the regression function directly [Cal16]. Instead, it models the joint probability density of the data and computes the motion from the density model. The time for regression is independent of this number and solely depends on the dimension of the data set and the number of mixture components. Thus, GMR is better suited for online motion synthesis than e.g., GPR, where the time for regression grows with the size of the data set. Secondly, as the underlying Gaussians are smooth functions, GMR inherently produces smooth and continuous motions.

In the context of learning adaptive task constraints for WBC, GMR has the advantage of directly providing information about the variability in the user demonstrations in terms of full covariance matrices. This information can be exploited to estimate the importance of individual constraints and relate it to the corresponding task priorities. Another advantage is that GMR achieves good generalization capabilities, even with a small number of demonstrations.

However, as GMR uses the means of the posterior probabilities to predict the motion, it will produce skills that average over the given user demonstrations. In case of having e.g., two different modes in the data set (two distinct clusters which are equally probable), this is an undesirable property, since none of the modes will be correctly represented. In this case,

an alternative to GMR is to sample on the joint probability distribution represented by the GMM [Bis06]. By providing a confidence interval that the sampled data point should be contained in, we can control the degree of how much averaging between different modes in the data set is allowed.

### Generalization to Unknown Contexts

Applying GMM-GMR to learn a demonstrated task has the purpose of achieving certain generalization capabilities. That is, the learned model is supposed to generalize with respect to novel, previously unseen situations. To achieve this, we perform user demonstrations in several different contexts, i.e., variations of the considered task. As already mentioned, we describe context as vector  $\kappa \in \mathbb{R}^C$  and map categorical variables to real-valued numbers by the means of simple one-hot encoding. Previous approaches for learning task constraints, like the one described by Calinon [Cal16], mostly generalize over start and/or goal positions for a given motion. The idea of introducing categories as context variables is that we want to generalize over more severe context changes. As an example, consider the task of carrying an object. Variations of the task may include changes in object geometry or weight, the way of task execution, e.g., allow/forbid tilting the object, and the robot morphology, e.g., executing the task with a single arm or with a dual-arm robot. In WBC such changes can be represented by an appropriate modification of the soft task priorities of certain tasks. For example, if an object may be tilted during execution, the soft task priorities related to the object orientation can be set to a low value, such that the robot can execute additional tasks instead, e.g., avoiding obstacles.

One problem with Gaussian Mixture Models is the selection of the number of components or mixtures. If it is chosen too large, the resulting model may represent the training data accurately, but does not generalize well to previously unseen samples. The variant of GMM that we employ here, DPGMM, allows to infer the number of active components from data, with the downside of requiring additional hyper-parameters to tune. The most important hyper-parameter is the weight concentration prior  $\gamma$ . A small value of  $\gamma$  sets most component's weights to zero, which leads to a small number of active components in practice. A large value of  $\gamma$  produces an equally distributed weight concentration over all components, which corresponds to having a many active components. To achieve the best generalization capabilities, we optimize the weight concentration prior and other hyper-parameters of the DPGMM using grid search. The training data is selected using leave-one-out cross validation. We use data from each context as a holdout set once and train on the data acquired in all remaining contexts. Finally, we test the resulting model in a context that the model has not seen before.

#### 4.5.3 Results

This section presents the experimental evaluation of the PbD-based approach for learning task constraints.



(a) *Rotate object*: Rotating an object by  $90^\circ$  degrees

(b) *Human-Robot Collaboration*: Collaborative transport of a bulky object

(c) *Assembly*: Connecting a tube and a connector piece

Figure 4.5: Kinesthetic teaching of dual-arm manipulation tasks, screenshots from video [MK21]

## Experimental Setup

The experiments are conducted on the iMRK system as described in Section 1.2. Initially, we must select suitable task frames. We select the base frame of the robot, denoted as *Base*, and the gripper frames, denoted as *Left EE* and *Right EE*. The corresponding task constraints are denoted as *Base-Left EE*, *Base-Right EE* and *Left EE-Right EE*. Having three 6-dimensional Cartesian task constraints, we get  $M = 18$  pose and twist variables, respectively. For each context, we perform  $D = 10$  user demonstrations with a varying start pose. The acquired task trajectories are resampled to  $S = 200$  samples. The robot model is based on the Unified Robot Description Format (URDF) [Sys21], which supports mechanical tree structures with serial chains of rigid bodies, connected by rotational and prismatic joints. It provides kinematic information to the underlying WBC implementation we use here (see Section 4.3).

Our approach is evaluated in 3 different manipulation tasks. Table 4.2 shows a summary of all contexts and context variables.

**Rotate Object** In this task the robot is supposed to rotate a rigid object by  $90^\circ$  (see Figure 4.5(a)). Variants of the task include the starting pose, the size of the object (the width is varied between  $0.3m$  and  $0.5m$ ), the direction of rotation (clockwise/anticlockwise) and the robot morphology (use of left arm, right arm or both arms). In total 14 different contexts are obtained, which can be described by  $C = 4$  context variables.

**Human-Robot Collaboration** Here, a human carries a bulky object in collaboration with the robot (see Figure 4.5(b)). Variants of the task include the starting pose, the way of task execution (allow/forbid tilting the object during transport) and the robot morphology (use of left arm, right arm or both arms). In total 6 different contexts are obtained, which can be described by  $C = 3$  context variables.

**Assembly** In this task the robot is supposed to assemble a tube and a connector piece (see Figure 4.5(c)). Variants of the task include the starting pose and the robot morphology (use of left arm, right arm or both arms). In total 3 different contexts are obtained, which can be described by  $C = 2$  context variables.

Rotate Object					
#	Description	OS	LA	RA	CW
$R_{11}$	Rotate obj. 0.30m clockwise	0.3	1	1	1
$R_{12}$	Rotate obj. 0.35m clockwise	0.35	1	1	1
$R_{13}$	Rotate obj. 0.40m clockwise	0.4	1	1	1
$R_{14}$	Rotate obj. 0.45m clockwise	0.45	1	1	1
$R_{15}$	Rotate obj. 0.50m clockwise	0.5	1	1	1
$R_{21}$	Rotate obj. 0.30m anticlockwise	0.3	1	1	0
$R_{22}$	Rotate obj. 0.35m anticlockwise	0.35	1	1	0
$R_{23}$	Rotate obj. 0.40m anticlockwise	0.4	1	1	0
$R_{24}$	Rotate obj. 0.45m anticlockwise	0.45	1	1	0
$R_{25}$	Rotate obj. 0.50m anticlockwise	0.5	1	1	0
$R_{31}$	Rotate obj. 0.50m left arm anticlockwise	0.5	1	0	0
$R_{32}$	Rotate obj. 0.50m left arm clockwise	0.5	1	0	1
$R_{41}$	Rotate obj. 0.50m right arm clockwise	0.5	0	1	1
$R_{42}$	Rotate obj. 0.50m right arm anticlockwise	0.5	0	1	0

Human-Robot Collaboration				
#	Name	AT	LA	RA
$H_{11}$	HRC no tilt	0	1	1
$H_{12}$	HRC with tilt	1	1	1
$H_{21}$	HRC no tilt left arm	0	1	0
$H_{22}$	HRC with tilt left arm	1	1	0
$H_{31}$	HRC no tilt right arm	0	1	0
$H_{32}$	HRC with tilt right arm	1	1	0

Assembly			
#	Name	LA	RA
$A_{11}$	Assembly	1	1
$A_{21}$	Assembly left arm	1	0
$A_{31}$	Assembly right arm	0	1

Table 4.2: Contexts and context variables used for experimental evaluation, OS - Object Size, CW - Clockwise rotation, LA/RA - Left Arm/Right Arm, AT - Allow Tilt

The experiments are illustrated in Figure 4.5, which shows screenshots from different videos [MK21].

### Reproduction of Task Constraints

In this section, we evaluate the capability of the approach to generalize with respect to previously unseen situations, i.e., different variants of the demonstrated task. These variants are described by the means of the context vector  $\kappa$ . As mentioned before, we fit a joint distribution  $p(\mathbf{v}, \mathbf{x}, \kappa)$  to context data and task constraints by using a DPGMM. We use a DPGMM implementation provided by Pedregosa et al. [Ped+11]. Regarding GMR we use the approach described by Fabisch [Fab21]. We select  $K = 50$  for the number of mixture

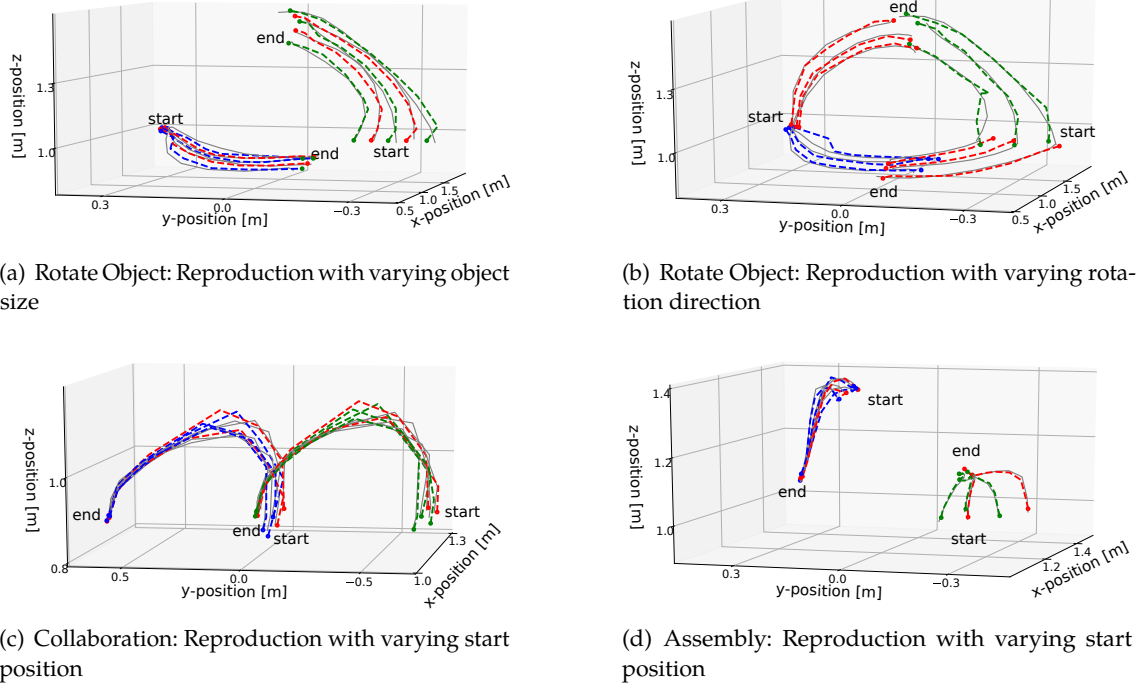


Figure 4.6: Results when reproducing task constraints in previously unseen context: Gray: Mean of demonstrations, Blue Dashed: Left Arm (constraint *Base-Left EE*), Green Dashed: Right Arm (constraint *Base-Right EE*), Red Dashed: Reproduction in previously unseen context

components and let the DPGMM compute the effective number of mixtures. Reproduction of the task constraints and soft task priorities is achieved as described in Algorithm 1.

The results are displayed in Figure 4.6 and explained in the following.

**Rotate Object** In the first case, the DPGMM is trained for clockwise rotation direction using both arms. We only use data obtained in the contexts  $\{R_{11}, R_{13}, R_{15}\}$  for model fitting. Figure 4.6(a) illustrates the reproduction results in the test contexts  $\{R_{12}, R_{14}\}$ , which correspond to object sizes, which are not contained in the training set. We only display the  $xyz$ -position for the sake of clarity. The DPGMM generalizes well over the size of the manipulated object. Figure 4.7(a) shows video snapshots of the experiment.

Secondly, the DPGMM is trained for clockwise rotation direction using both arms and anticlockwise rotation direction with only a single arm. Thus, we only use data obtained in the contexts  $\{R_{11} \dots R_{15}, R_{31}, R_{32}, R_{41}, R_{42}\}$  for training. The learned model is evaluated using *anticlockwise* rotation using *both arms*. This variant of the task corresponds to the contexts  $\{R_{21}, \dots, R_{25}\}$ , which are unknown to the learned model. The results are illustrated in Figure 4.6(b). The DPGMM is able to reproduce the task with both hands in anticlockwise rotation direction, a variant that is not included in the training set.

**Human-Robot Collaboration** In this case, the model is trained using only a part of the demonstrations ( $D = 6$ ) for a fixed context ( $H_{11}$ ). Each of the demonstrations has a different starting pose. The remaining demonstrations ( $D = 4$ ) with unknown starting poses are used for evaluation. The results are illustrated in Figure 4.6(c). The model is able to generalize



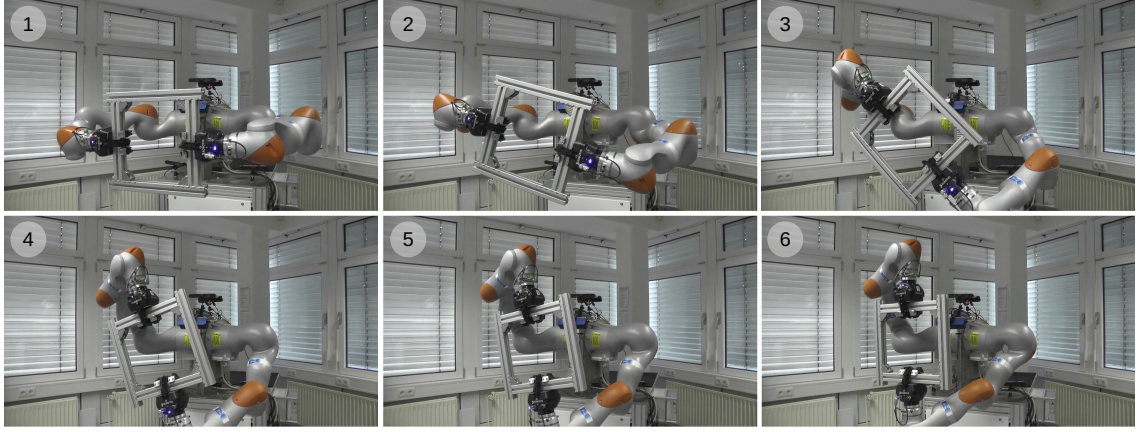
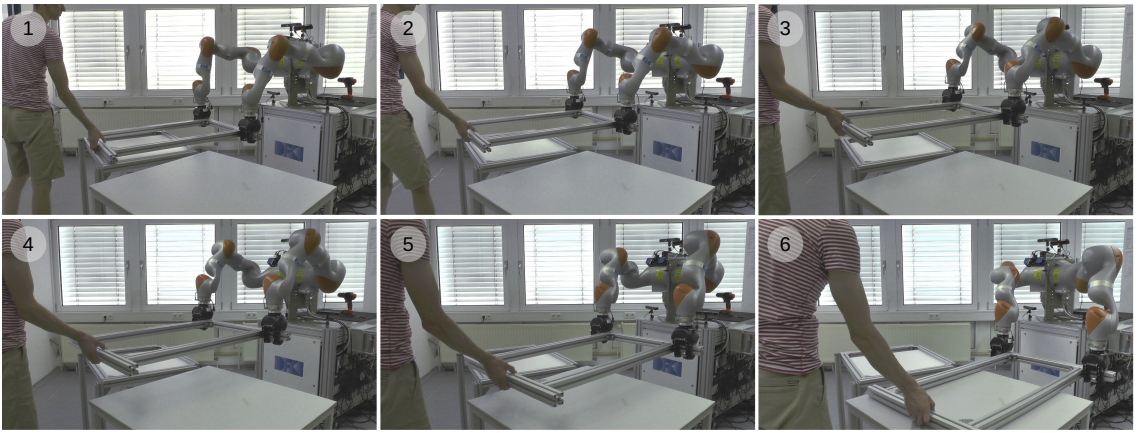
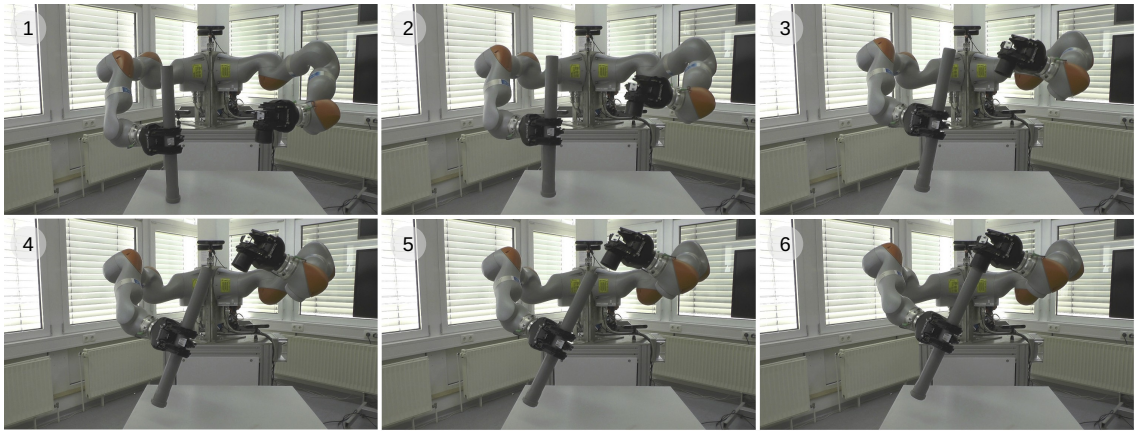
(a) Reproduction of the *Rotate Object* task in context  $R_{12}$ (b) Reproduction of the *Human-Robot Collaboration* task in context  $H_{11}$ (c) Reproduction of the *Assembly* task in context  $A_{11}$ 

Figure 4.7: Results on reproduction of task constraints, screenshots from video [MK21]

over varying starting poses. Like before, we only show the  $x/y/z$ -position. Figure 4.7(b) shows video snapshots of the experiment.

**Assembly** Finally, the DPGMM is trained with  $D = 6$  demonstrations of the assembly task. Each demonstration has a different starting pose. We use a fixed context  $A_{11}$  here. The remaining  $D = 4$  demonstrations with unknown starting poses are used for evaluation. The

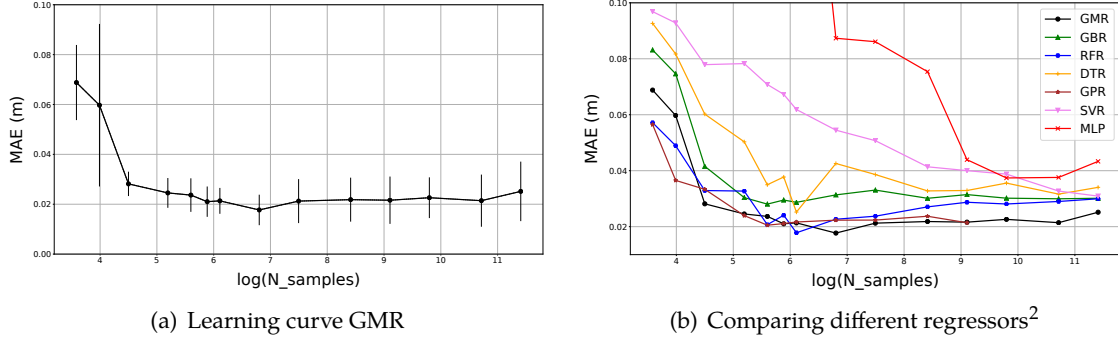


Figure 4.8: Learning curves for the *Rotate Object* task, x-axis: Number of training samples, y-axis: mean absolute reproduction error for the contexts  $R_{11} \dots R_{15}$  and  $R_{21} \dots R_{25}$

results are displayed in Figure 4.6(d). Again, the approach can generalize over previously unknown starting poses. Figure 4.7(c) shows video snapshots of the experiment.

**Model Performance** We analyze the performance of our approach using the contexts ( $R_{11} \dots R_{15}$ ) and ( $R_{21} \dots R_{25}$ ) of the *Rotate Object* task (see Table 4.2). The model is trained with all but one of these contexts using grid search and leave-one-out cross validation for hyper-parameter tuning. Then we evaluate the resulting model by measuring the error between the mean of the demonstrations and the reproduced motion (predictions made by the model) for the remaining (unknown) context. Every context is used once for evaluation, so we perform  $C$  evaluations in total. As a performance measure we use the mean absolute error (MAE) over all evaluations.

Figure 4.8(a) shows the learning curve (number of training samples vs. MAE including a single standard deviation depicted as error bars) for the *rotate object* task. The motion can be reproduced reliably in unknown contexts with low error (approx. 2cm on average). In Figure 4.8(b) we compare the GMR learning curve with other regression methods used for motion synthesis. It shows that GMR requires a small number of training samples to achieve good generalization capabilities (low reproduction error in unknown contexts) and provides a low overall reproduction error in general.

### Estimation of Soft Task Priorities

In this section, we evaluate the ability of the approach to estimate soft task priorities and adapt them

1. in a temporal manner, i.e., during task execution
2. with respect to different aspects of the tasks, e.g., different task variables receive different soft task priorities depending on their importance
3. according to the current context

As described in Section 4.5, we estimate the task priorities from the variance in the user demonstrations according to Equation (4.10).

<sup>2</sup> GMR - Gaussian Mixture Regression, GBR - Gradient Boosting Regression, RFR - Random Forest Regression, DTR - Decision Tree Regression, GPR - Gaussian Process Regression, SVR - Support Vector Regression, MLP - Multi-layer Perceptron, All implementations are taken from scikit-learn [Ped+11]

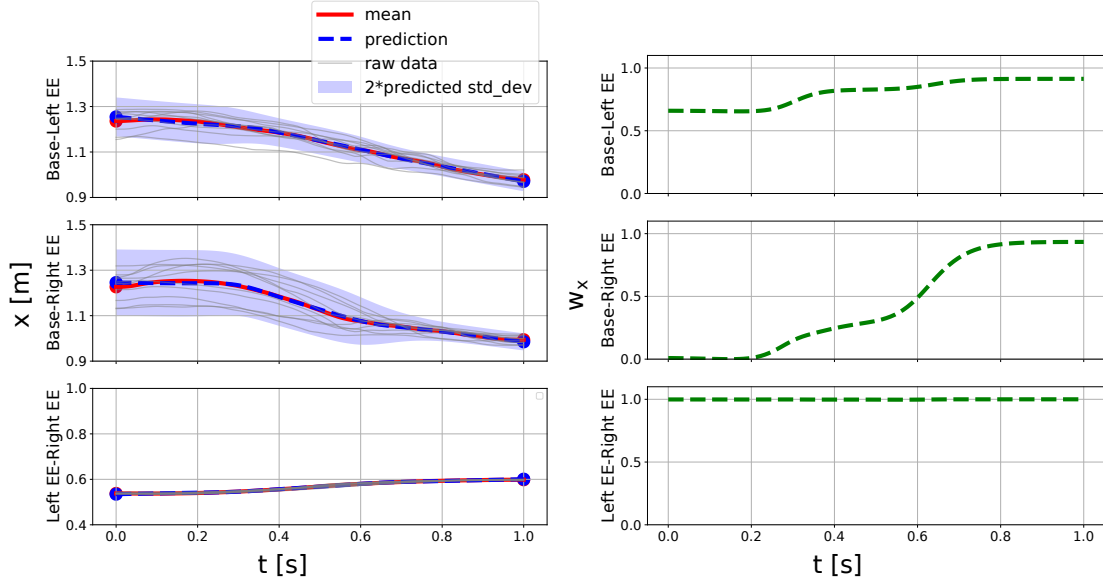
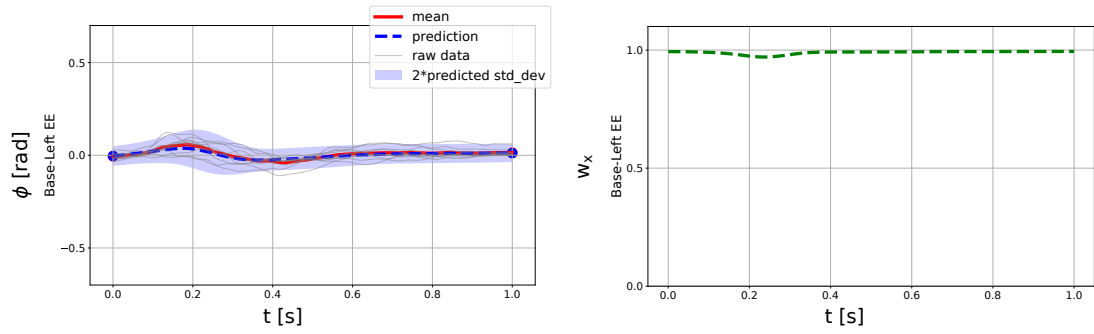
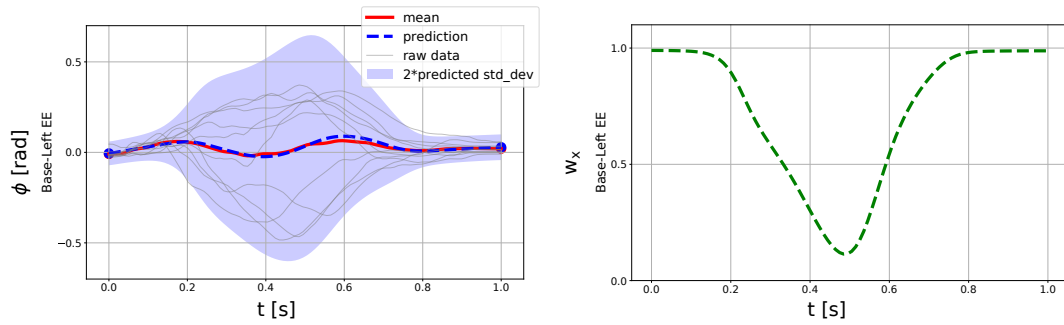
(a) Rotate Object task (only  $x$ -position, fixed context  $R_{15}$ ).(b) Human-Robot Collaboration task (only  $\phi$ -orientation), Reproduction in context  $H_{11}$ : Without tilting(c) Human-Robot Collaboration task (only  $\phi$ -orientation), Reproduction in context  $H_{12}$ : With tilting

Figure 4.9: Estimation of soft task priorities: Temporal, inter-constraint and contextual adaptation. *Left*: Reproduction of task constraints and variance, *Right*: Estimation of task priorities according to (4.10).

**Temporal & Inter-Constraint Adaptation** We use the *Rotate Object* task to illustrate the ability of temporal and inter-constraint adaptation of the approach. Figure 4.9(a) shows the



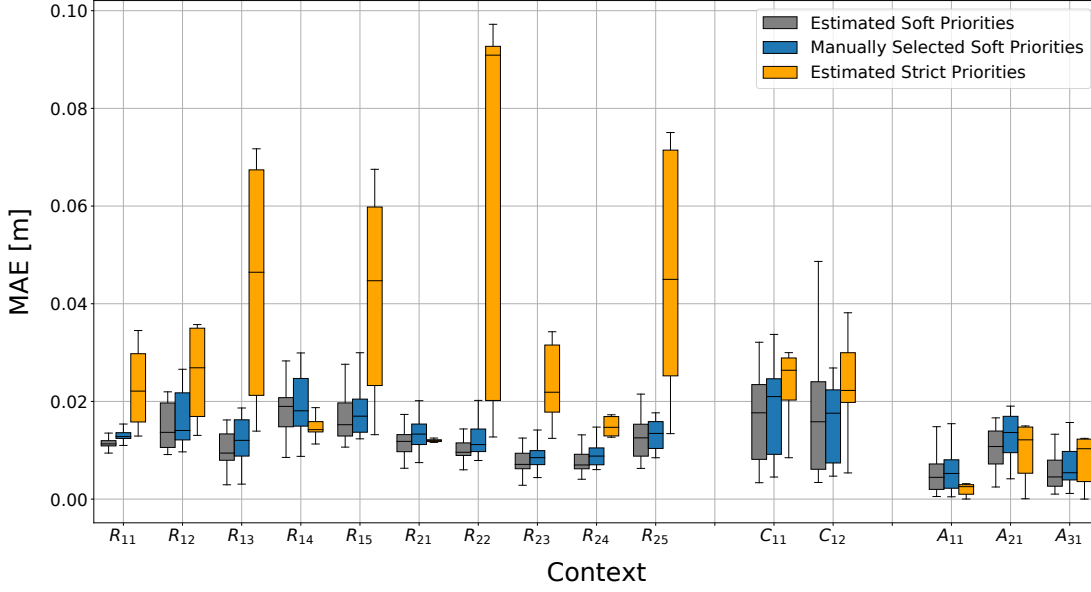


Figure 4.10: Comparison of the reproduction error using three different methods for task prioritization: Estimated soft task priorities (approach described in Section 4.5), manually selected soft task priorities (blue) and estimated strict hierarchies as proposed by Silv rio et al. [Sil+19] (orange).

reproduction of this task (only x-axis), the user demonstrations, their mean and the estimated confidence interval  $2\sigma$ . Since the demonstrations are performed from different starting poses, the result initially shows a large variance for *Base-Left EE* and *Base-Right EE*. This variance becomes smaller during task execution, since the object is moved to a similar end pose in each demonstration. The corresponding task priorities are low in the beginning of the motion and increase towards the end. In contrast, the constraint *Left EE - Right EE* has a low variance and, accordingly, a large task priority during the entire motion. The reason is that the relative motion of the grippers is constrained by the rigid object that they are carrying. The resulting soft task priorities reflect the importance of the different constraints. In this case, the relative pose of the grippers is more important than the pose of each individual gripper. Also, it is shown that the soft task priorities are adapted correctly during task execution.

**Context Adaptation** Figures 4.9(b) and 4.9(c) show the effect of estimating soft task priorities in different contexts for the *Human-Robot Collaboration* task (only  $\phi$ -rotation). For evaluation, we use the contexts  $H_{11}$  (Figure 4.9(b)) and  $H_{12}$  (Figure 4.9(c)). Since we allow tilting the carried object during the motion in context  $H_{12}$ , the variance is large, and the corresponding task priority drops during task execution. In contrast, the task priority in context  $H_{11}$  is high throughout the whole motion. A video demonstrating the contextual adaptation of the task priorities can be found with the related publication [MK21].

#### 4.5.4 Comparison of Different Approaches for Task Prioritization

In this section we compare three different approaches for task prioritization:

- i. Our approach for estimating soft task priorities using GMR as described in Section 4.5.

- ii. Manually selected soft task priorities. Here we select a fixed value  $w = 1$  for all task priorities during the complete motion.
- iii. Strict hierarchy estimation from data as described by Silv rio et al. [Sil+19].

The latter approach enforces strict prioritization through consecutive null space projections of the respective task Jacobians. The task hierarchy is thereby estimated from the variance in the user demonstrations, given a set of candidate hierarchies. The idea is that hierarchies with a low relevance produce a higher variability during demonstration than hierarchies with high relevance. For movement synthesis the candidate joint space velocities generated by each hierarchy are fused using a soft weighting scheme, where a high variance in the user demonstrations corresponds to a low priority and vice versa.

For evaluation, we train the models using all but one of the contexts of a given task and measure the reproduction error for the remaining, previously unknown context. Again, we consider three task constraints, denoted as *Base-Left EE (left)*, *Base-Right EE (right)* and *Left EE-Right EE (relative)*. For the approach described by Silv rio et al. [Sil+19], we must select a set of candidate hierarchies in advance. For simplicity, we select each combination, given the three task constraints:

priority	highest	medium	lowest
$h_1$	<i>left</i>	<i>right</i>	<i>relative</i>
$h_2$	<i>left</i>	<i>relative</i>	<i>right</i>
$h_3$	<i>right</i>	<i>left</i>	<i>relative</i>
$h_4$	<i>right</i>	<i>relative</i>	<i>left</i>
$h_5$	<i>relative</i>	<i>right</i>	<i>left</i>
$h_6$	<i>relative</i>	<i>left</i>	<i>right</i>

The evaluation results are shown in Figure 4.10. As before, we evaluate the MAE between the mean of the user demonstrations and the predicted motion in different contexts.

From these figures we can derive the following results: (1) Our approach for estimating soft task priorities results in a lower reproduction error compared to the use of fixed task priorities. Apart from that, it allows a bigger flexibility for executing additional tasks like e.g., collision avoidance. (2) Our approach results in a lower reproduction error when comparing to the method described by Silv rio et al. [Sil+19] (fusion of strict hierarchies). This is due to the following reasons: In all three evaluation tasks, we have an over-constrained case (18 constraints, but only 14 degrees of freedom). Obviously, strict prioritization is not useful in such a case since it results in a bad tracking performance for the tasks with lowest priority. Fusing the candidate hierarchies with a soft weighting scheme cannot overcome this issue, at least not with the training data that we acquired. (3) The mean reproduction error of our method is in the magnitude of around  $0.005m - 0.02m$ . Since we are not regarding high precision tasks here and the KUKA robots have integrated compliance controllers that compensate for minor inaccuracies, the resulting reproduction errors are acceptable to us. Further improvements can be made by obtaining more examples from user demonstrations.

## 4.6 Alternative Encoding: Probabilistic Movement Primitives

In this section, we describe an alternative approach for encoding contexts and task constraints based on Probabilistic Movement Primitives (ProMPs). We adapt the approach described in the previous section regarding two aspects. Firstly, we employ ProMPs as defined in (2.28) as motion model. To represent the actual trajectory, we use a mixture of radial basis functions (RBFs) in the ProMP. Secondly, we apply a hierarchical approach for contextual adaptation. As ProMPs cannot directly encode context variables, we use GMMs to model the mapping from context to ProMP weights. From the estimated weight vector, we can reproduce mean, and variance of the movement given the initial position  $\mathbf{x}_0$  of all task constraints. ProMPs have similar characteristics like ordinary DMPs, that is they can be conditioned according to a new starting or ending position and allow temporal scaling. Apart from that, when being learned from multiple demonstrations, ProMPs provide information about the variability of the demonstrated trajectories in terms of full covariance matrices.

### 4.6.1 Probabilistic Encoding and Reproduction of Task Constraints

As in Section 4.5, we want to estimate the task constraints  $(\mathbf{x}(t), \mathbf{v}(t))$  along with the associated soft task priorities  $\mathbf{w}(t)$  for a certain context. Given a training data set  $\xi = \{\mathbf{K}, \mathbf{X}, \mathbf{V}\}$  obtained from user demonstrations, we derive a probabilistic model that is supposed to encode context and task constraints. In contrast to the approach in Section 4.5.1, we first fit a ProMP to each demonstration and obtain the corresponding ProMP weights  $\mathbf{w}$ . The weights are computed using a linear regression over the poses in the demonstrated trajectory. From the new data set, consisting of context data and ProMP weights, we model the joint probability distribution:

$$p(\boldsymbol{\kappa}, \mathbf{w}) = \sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{\kappa}, \mathbf{w} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (4.11)$$

Again,  $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$  are the model parameters denoted as mixing weights, means and covariance matrices of the Gaussian distributions  $\mathcal{N}(\boldsymbol{\kappa}, \mathbf{w})$ . From the conditional distribution  $p(\mathbf{w} | \boldsymbol{\kappa})$ , we can now obtain the ProMP weights  $\mathbf{w}$  for a certain context  $\boldsymbol{\kappa}$  by using Gaussian Mixture Regression. Having computed the ProMP weights, we can compute the probability of observing the task constraints  $(\mathbf{x}(t), \mathbf{v}(t))$  given an initial pose  $\mathbf{x}_0$  as:

$$p(\mathbf{x}_t, \mathbf{v}_t | \mathbf{w}, \mathbf{x}_0) = \prod_t \mathcal{N}(\mathbf{x}_t, \mathbf{v}_t | \boldsymbol{\Phi}_t^T \mathbf{w}, \boldsymbol{\Sigma}_t) \quad (4.12)$$

where  $\boldsymbol{\Phi}_t$  is the RBF matrix. Here, we apply RBFs of the form  $\phi(r) = \exp(a(r - c)^2)$ , where the centers  $c$  are equally distributed in the interval  $[0, 1]$  and the factor  $a$  is chosen such that the RBFs overlap to a certain degree. The mean values of this probability distribution reproduce the task constraints in the context  $\boldsymbol{\kappa}$  and for the initial pose  $\mathbf{x}_0$ . From the trajectory variance, we can get an estimate of the soft task priorities according to (4.10).

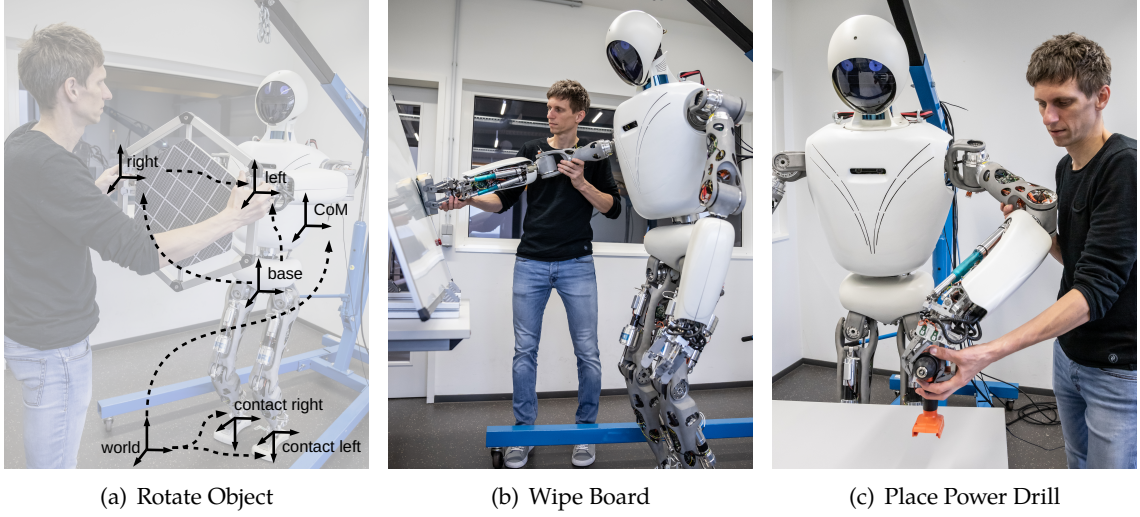


Figure 4.11: Kinesthetic teaching on the RH5 humanoid. Illustration of task frames and constraints.

#### 4.6.2 Results

##### Experimental Setup

We evaluate the ProMP-based approach for estimating task constraints and their associated priorities on the humanoid robot RH5 shown in Figure 1.3. As WBC approach, we employ a variant of the QP in (2.15), which includes rigid contact constraints for the RH5 floating base model:

$$\begin{aligned}
 & \underset{\dot{\mathbf{q}}}{\text{minimize}} && \left\| \sum_{i=1}^P (\mathbf{J}_{W,i} \dot{\mathbf{q}} - \mathbf{v}_{d,i}) \right\|_2 \\
 & \text{subject to} && \mathbf{J}_c^j \dot{\mathbf{q}} = \mathbf{0}, \quad \forall j \\
 & && \mathbf{J}_{com} \dot{\mathbf{q}} = \mathbf{v}_{d,com}
 \end{aligned} \tag{4.13}$$

Again,  $\mathbf{J}_{W,i}$  is the weighted Jacobian and  $\mathbf{v}_{d,i}$  the desired spatial velocity of task  $i$ , respectively. Each spatial velocity is generated by a Cartesian pose controller as in (4.2). The constraint  $\mathbf{J}_c^j \dot{\mathbf{q}} = \mathbf{0}$  ensures rigid, non-moving contacts, where  $\mathbf{J}_c^j$  is the contact Jacobian of contact point  $j$ . In this case we have at least two rigid contacts of the feet with the ground floor,  $j = \{1, 2\}$ . The constraint  $\mathbf{J}_{com} \dot{\mathbf{q}} = \mathbf{v}_{d,com}$  ensures that the CoM remains within the support polygon of the robot feet, such that the overall balance of the system is maintained. In contrast to (4.1), we formulate the task constraints, which we attempt to learn from user demonstrations, within the cost functional of the QP. In doing so, we combine expert knowledge considering the appropriate specification of the CoM controller with task constraints learned from demonstration. Figure 4.11(a) illustrates the task frames and constraints used for experimental evaluation.

As in Section 4.5.3, we perform kinesthetic teaching to record training data for different tasks as illustrated in Figure 4.11. We vary the context in between the demonstrations, i.e., we perform the experiments considering different variants of the tasks. For each context, we perform  $D = 10$  user demonstrations, in which we record the task constraints  $(\mathbf{x}(t), \mathbf{v}(t))$  for

Rotate Panel							
#	Description	RA	CW				
$R_{11}$	Rotate Panel 30° clockwise	0.5236	1				
$R_{12}$	Rotate Panel 60° clockwise	1.0472	1				
$R_{21}$	Rotate Panel 30° anticlockwise	0.5236	0				
$R_{22}$	Rotate Panel 60° anticlockwise	1.0472	0				

Wipe Whiteboard							
#	Name	WA	CW	AW	HO	LA	RA
$W_{11}$	Wipe Board 40° clockwise left	0.6981	1	0	0	1	0
$W_{12}$	Wipe Board 50° clockwise left	0.8727	1	0	0	1	0
$W_{13}$	Wipe Board 60° clockwise left	1.0472	1	0	0	1	0
$W_{14}$	Wipe Board 70° clockwise left	1.2217	1	0	0	1	0
$W_{15}$	Wipe Board 80° clockwise left	1.3962	1	0	0	1	0
$W_{21}$	Wipe Board 40° anticlockwise left	0.6981	0	1	0	1	0
$W_{21}$	Wipe Board 40° horizontal left	0.6981	0	0	1	1	0
$W_{31}$	Wipe Board 40° clockwise right	0.6981	1	0	0	0	1
$W_{32}$	Wipe Board 40° anticlockwise right	0.6981	0	1	0	0	1
$W_{33}$	Wipe Board 40° horizontal right	0.6981	0	0	1	0	1

Place Object				
#	Name	TH	LA	RA
$P_{11}$	Place Object 77cm left	0.77	1	0
$P_{12}$	Place Object 80cm left	0.80	1	0
$P_{13}$	Place Object 83cm left	0.83	1	0
$P_{14}$	Place Object 85cm left	0.85	1	0
$P_{15}$	Place Object 87cm left	0.87	1	0
$P_{21}$	Place Object 77cm right	0.77	0	1
$P_{22}$	Place Object 80cm right	0.80	0	1
$P_{23}$	Place Object 83cm right	0.83	0	1
$P_{24}$	Place Object 85cm right	0.85	0	1
$P_{25}$	Place Object 87cm right	0.87	0	1

Table 4.3: Tasks and context variables used for experimental evaluation on RH5, RA - Rotation angle, WA - Whiteboard Angle, CW - Clockwise, AW - Anticlockwise, HO - Horizontal, LA/RA - Left Arm/Right Arm, TH - Table Height

each pair of the task frames *left*, *right*, *base* and *world*. The resulting trajectories are resampled to contain  $S = 200$  samples each.

The following tasks are used for experimental evaluation (see Table 4.3 for an overview):

**Rotate Panel** The RH5 humanoid rotates a solar panel with two arms (Figure 4.11(a)). The task is taken from the storyline of the TransFit project<sup>3</sup>. We vary the rotation angle (30°, 60°) and the rotation direction (clockwise, anticlockwise), such that we get  $C = 2$  context variables and 4 different contexts in total.

<sup>3</sup> The TransFit project is funded by the German Federal Ministry of Economic Affairs and Energy (BMWi) as well as the German Aerospace Center (DLR), Grant number 50RA1701

**Wipe Whiteboard** The robot is supposed to wipe a whiteboard using a whiteboard cleaning sponge (Figure 4.11(b)). We vary the tilt of the whiteboard ( $40^\circ - 80^\circ$ ), the type of wiping motion (circular motion clockwise/anticlockwise, horizontal motion) and the arm to be used (right/left arm). We get  $C = 6$  context variables and record 10 different contexts.

**Place Object** The robot is supposed to place an object on the table with one hand, while holding to a vertical bar with the other hand (Figure 4.11(c)). We vary the height of the table ( $77cm - 87cm$ ) and the arm to be used for placing the object. We get  $C = 2$  context variables and we record 10 different contexts here.

### Reproduction of Task Constraints

From the data obtained in user demonstrations as illustrated in Figure 4.11 we learn a hierarchical model as described in Section 4.6.1. The task constraints ( $\mathbf{x}(t)$ ,  $\mathbf{v}(t)$ ) are represented as ProMPs. For each demonstration, we fit a ProMP to the task constraints and obtain the corresponding ProMP weights. Then, we model context data and ProMP weights as an upper level GMM with  $K = 10$  mixture components. From this model, we can obtain the ProMP weights for a given context using GMR. The approach also allows us to generalize with respect to previously unseen contexts. From the ProMP described by the estimated ProMP weights  $\hat{\mathbf{w}}$ , we obtain the task constraints ( $\hat{\mathbf{x}}(t)$ ,  $\hat{\mathbf{v}}(t)$ ) as well as the soft task priorities  $\hat{\mathbf{w}}(t)$  according to (4.10).

Using this approach, we reproduce the tasks *Rotate Panel*, *Wipe Whiteboard* and *Place Object* on the RH5 humanoid (see Figure 4.12). In all three cases RH5 is standing freely, integrating manually specified CoM-control with the learned task constraints in order to safely maintain balance while manipulating with one or two arms. Figure 4.13 shows the reproduced task trajectories (only left arm position). In Figure 4.13(a), the results on the *Wipe Whiteboard* tasks are shown. The approach can generalize with respect to a varying inclination of the whiteboard with respect to the floor. While we train the hierarchical model with the contexts  $W_{11}$ ,  $W_{13}$ ,  $W_{15}$  corresponding to inclinations of  $40^\circ$ ,  $60^\circ$ ,  $80^\circ$  (blue solid lines), we evaluate the model in the contexts  $W_{12}$ ,  $W_{14}$ , which correspond to inclinations of  $50^\circ$ ,  $70^\circ$  (red dashed lines). Figure 4.13(b) shows the results on the *Place Object* task. As the power drill is too heavy to allow safe manipulation with the RH5 grippers, we evaluate the task with a box of the same height as the power drill. The plot shows that we can generalize the task constraints with respect to varying table heights, which correspond to varying target locations of the handled object. Again, we train the model in several contexts, in this case  $P_{12}$ ,  $P_{13}$ ,  $P_{15}$  corresponding to table heights of  $80cm$ ,  $83cm$ ,  $87cm$ , and evaluate it with the previously unknown context  $P_{14}$ , which corresponds to the table height  $85cm$ .

### Comparison to GMM-GMR

In this section, we compare the hierarchical, ProMP-based method with the end-to-end approach using GMM-GMR as described in Section 4.5.3. As training data, we use the *Rotate Object* data set obtained by kinesthetic teaching on the KUKA dual arm system (see Figure 4.5(a)). We only consider the contexts  $R_{11}, \dots, R_{15}, R_{21}, \dots, R_{25}$  (see Table 4.2) for training. Both models are trained using grid search and leave-one-out cross validation for hyper-parameter tuning. The most important hyper-parameter for the GMM is the number



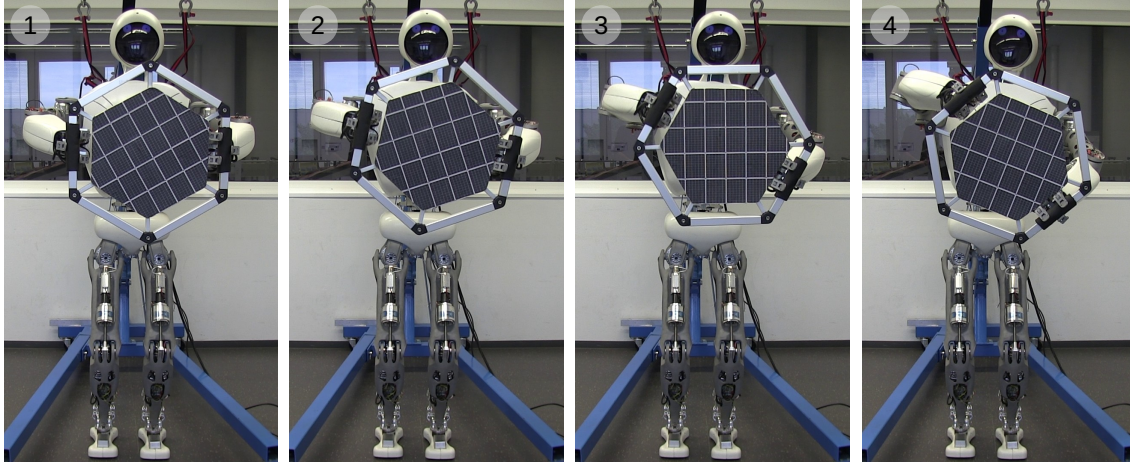
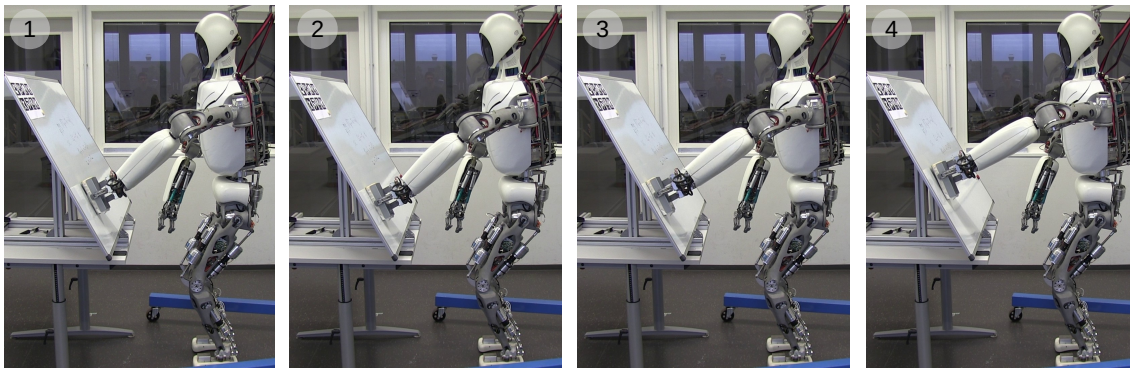
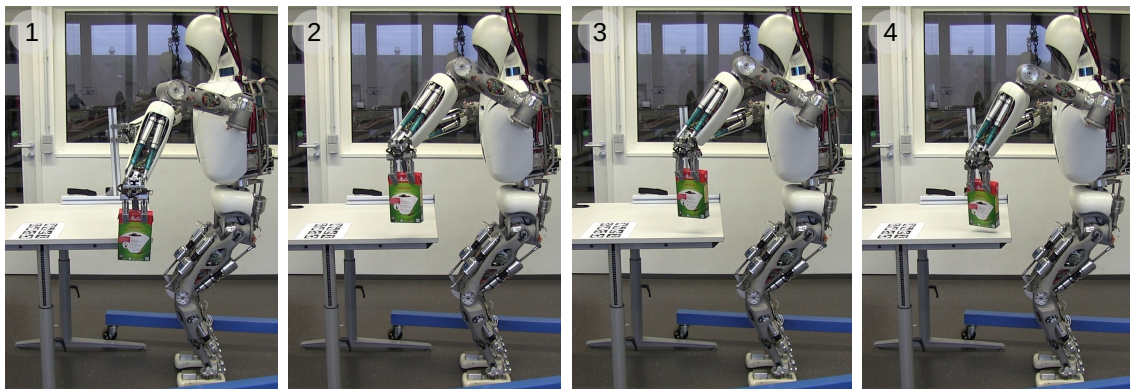
(a) Rotate Panel, Context  $R_{12}$ (b) Wipe Whiteboard, Context  $W_{13}$ (c) Place Object, Context  $P_{13}$ 

Figure 4.12: Reproduction of task constraints using the ProMP-based approach

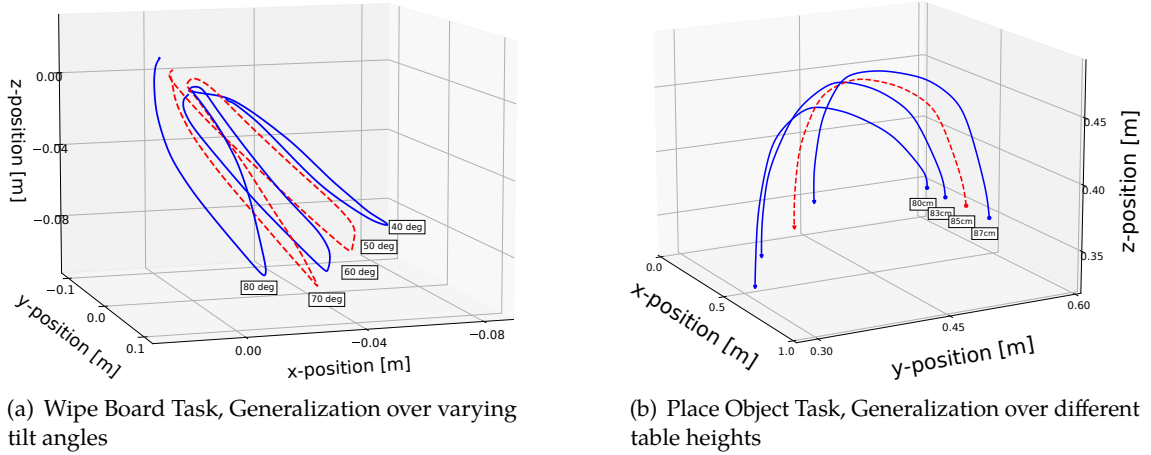


Figure 4.13: Results on reproduction of task constraints on the RH5 robot, left arm position. **Blue solid:** Known contexts, **Red dashed:** Reproduction in unknown context.

of mixture components. For the ProMP-based approach, we have the number of GMM components for the context mapping on the one hand and the number of basis functions for shaping the actual movement primitive on the other. The shape and parameters of the RBFs remain fixed during grid search. For evaluation, we train the models with all but one of the given contexts and measure the MAE between the demonstrations and the reproduced motion for the remaining, previously unseen context. Every context is used once for evaluation, so we perform 10 evaluations in total. Figure 4.14 shows the results of this evaluation. Note that we only show the accuracy on the position variables here and leave out the orientation values. However, the rotational error between demonstration and reproduced motion shows similar results.

Both approaches perform similarly on average. We obtain an overall MAE of  $0.0269 \pm 0.0116\text{m}$  for GMM-GMR and  $0.0270 \pm 0.0151\text{m}$  for the ProMP-based approach. However, the GMM-GMR approach requires more time for fitting and prediction. Especially the prediction time could be a burden for real-time applications. We found that the ProMP-based approach requires  $0.5 - 1\text{ ms}$  for predicting the next data point on average, while GMM-GMR is slower by a factor of 10 when using the optimal set of hyper-parameters acquired through grid search. A disadvantage of the ProMP-based approach is that it can only be applied if the context is constant throughout the entire task execution. If the context suddenly changes, a new set of ProMP weights must be acquired, which brings additional computational costs. Furthermore, a smooth task space trajectory cannot be ensured in this case.



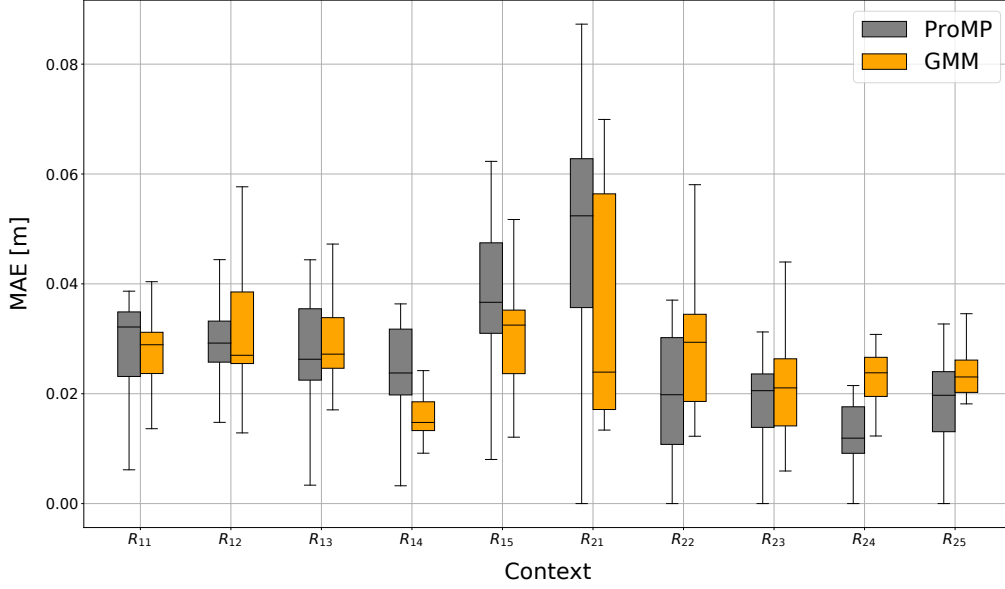


Figure 4.14: Comparison of the GMM-GMR and the ProMP-based approach for the *Rotate Object* data set obtained on the KUKA dual-arm robot.

## 4.7 Discussion

In this chapter, we introduce a PbD-based approach to derive task constraints and associated priorities of a whole-body controller. We found that the use of PbD in combination with WBC offers an intuitive way to program multiple simultaneous tasks on redundant robots, reducing the effective amount of time spent with manual tuning and evaluation of task specifications. Furthermore, the introduced methods can generalize task constraints and priorities over context changes using Gaussian Mixture Regression. As a result, we achieve an improved performance compared to manual tuning and the human expert is relieved from the burden of having to reprogram the task for every novel situation. Instead, the generalization capabilities of the model can be exploited. Furthermore, we showed that the developed approach outperforms a state-of-the-art method, which relies on a weighted mixture of strict priorities [Sil+19].

On the downside, the PbD-based approach described in this chapter obviously relies on the quality of user demonstrations. These must reflect the constraints of the demonstrated task. If the user demonstrations do not cover the important aspects of the task, the estimated task priorities might be suboptimal, e.g., the solution might be unnecessarily over- or under-constrained. For example, when teaching a robot to place an object on a table, its motion is obviously constrained in the direction perpendicular to the surface (z-direction), while it can place the object quite freely somewhere on the table (xy-direction). The user demonstrations should address this by varying the target position on the surface as much as possible. This results in a high variance in xy-direction and a low priority for the corresponding constraints according to (4.10). Conversely, the z-direction is assigned a low variance and a high priority. This way, the resulting task priorities will allow the robot to fulfill additional tasks in xy-direction like e.g., obstacle avoidance. This simple example shows that a thorough design of the experiments for data acquisition is a crucial part of the approach. Active learning methods can be useful here to extend the programming-by-demonstration paradigm and

supply the user with hints on "what to teach next" (see e.g., the work presented by Calinon and Billard [CB07]).

In the developed PbD workflow there are still some manual steps to consider. Firstly, the user must choose the relevant task frames in advance. Although this choice is trivial in many cases, there might be some tasks where a certain expertise is required. In the proposed workflow, the number of task constraints is related to the number of combinations of the selected task frames. As the computational effort quickly grows with the number of constraints, the task frames should be selected with care. As an additional improvement, the information whether a task frame introduces redundant information on the task can be derived from data. This way, irrelevant task frames can be discarded before fitting the model. Secondly, the identification of the current context must be done by the user in a manual fashion. An automatic identification of the current context is out of the scope of this thesis. Automatizing this step requires classifying the current context based on sensor data of the robot, given the experience of previously demonstrated tasks. In addition, the system should be able to distinguish a novel, previously unseen context from the known ones. Both problems are considerable challenges in real-world applications.

Throughout this thesis, we partly use categorical variables to represent context in order to ease labeling of the demonstrations. Although Gaussians are usually not optimal for modeling categories, it could be shown that GMMs are well enough able to fit categorical context data if a suitable regularization of the model parameters is done. In future, it is worth investigating different representations of the categorical variables like, e.g., binomial distributions.

In the presented approach, we focus on estimation of soft task priorities. It can be shown that reasonable soft priorities can be estimated from data, which allows temporal and contextual adaptation of the task constraints, while providing a means to automatically identify the relative importance of the demonstrated subtasks. However, there are situations where task hierarchies provide better results than soft task priorities. Especially in the over-constrained case, where the number of task variables is larger than the number of robot dof, there may be subtle differences. When using soft task prioritization in an over-constrained case, a residual error will remain. The distribution of this error over the tasks depends on the choice of soft task priorities. If all priorities are set to the same value, the error will equally distribute over all tasks. When using task hierarchies, the higher prioritized tasks will produce zero residual error, while the task with lowest priority will show the largest error. However, when the desired control actions of the individual tasks are orthogonal, hierarchical controllers can exactly resolve over-constrained cases and produce zero error in task space. To allow the estimation of both, strict and soft task priorities, as required by hybrid WBC approaches, one future research direction will be to extend the presented approach to task hierarchies.

The user demonstrations provided may be suboptimal, i.e., they might not represent the structure of the overall task correctly. Consequently, the estimated task constraints and priorities might as well be suboptimal. Thus, we present optimization approaches for task priorities in the following chapter, which are meant to improve the performance of the whole-body controller when deployed on the target system.

In this chapter we present methods to optimize task priorities used in WBC. The goal is to improve the performance of a whole-body controller when deployed on the target robotic system. We focus on soft task priorities, which may be constant during task execution or a function of time. As we do not have prior knowledge about the shape of the cost function, we apply black-box optimization methods. Optimization may take place in simulation or on the actual robot. We evaluate the proposed methods on the iMRK robot illustrated in Figure 1.3.

Parts of this chapter have already been published [Mro+20; Gea+17a]

The original contributions presented in this chapter are

- i. A novel method for human-robot collision avoidance using WBC
- ii. An approach to automatically derive optimal control parameters and soft task priorities within a WBC framework based on black-box optimization, which provides improved control performance compared to manual tuning
- iii. A comparison of different cost functions for black-box optimization of task priority functions in WBC.

This chapter is organized into two main sections. Section 5.1 presents an approach for black-box optimization to automatize selection of time-fixed soft task priorities and other WBC control parameters. The approach is evaluated in a human-robot coexistence scenario with the focus on real-time collision avoidance. In Section 5.2 we consider optimization of task priorities as a function of time. Starting with the results obtained in Section 4.5.3, we apply black-box optimization to enhance the resulting soft task priority functions. We evaluate the approach on an industrial dual-arm robot and compare different cost functions used in literature.

## 5.1 Optimizing Task Constraints in Human-Robot Collision Avoidance

In this section, we describe an approach for safe human-robot coexistence based on WBC. We automatically derive optimal task parameters for the whole-body controller using black-box optimization. The approach improves the control performance in transient zones between constrained and unconstrained motion in terms of smoothness.

### 5.1.1 Motivation

Safe collaboration of humans and robots is of major interest in industrial manufacturing. The increasing number of collaborative lightweight robots applied in industry allows (to some extent) the removal of fences usually separating robots and workers. The use of modern sensor technologies and processing approaches allows external monitoring of collaborative workspaces and tracking of the human worker inside the dangerous zone. By maintaining a

minimal distance to the robot, the safety of the worker can be ensured. Existing approaches in the industrial domain usually perform a full stop of the robot as soon as the sensors detect a violation of the minimum distance. However, in order to enable closer collaboration as, e.g., in shared assembly and reduce downtime, more intelligent systems are required. These systems must enable robots to avoid unintended collisions and find escape trajectories, if possible, while only pausing the motion if a collision cannot be prevented.

Whole-Body Control provides a flexible approach to integrate the generation of avoidance motions with other robotic manipulation tasks. The robot is supposed to avoid collisions with arbitrary external objects in its workspace, while at the same time continue executing its main task, e.g., picking and placing objects, and additional secondary tasks with lower priority. Obstacles can be static, like a tool or a work piece that has been placed in the workspace, or dynamic, like a human walking into the reachable area of the robot. As already described in previous sections, WBC offers a powerful tool to specify and control complex robot tasks. However, the solution is usually controlled by many parameters like control gains and task priorities. In particular, the manual tuning of task priorities is time-consuming and prone to errors. Thus, we investigate automatized methods based on black-box optimization to identify WBC parameters in this section. We apply evolutionary methods for parameter optimization and define a suitable fitness criterion for optimization. Although the focus is on optimization of task priorities for WBC, we also introduce and discuss methods on collision distance computation and avoidance control. Collision distance computation is based on the Kinematic Continuous Collision Detection (KCCD) library for robotic self-collision avoidance [TBF11; TF12]. We extend the library to allow distance computations between robots and unknown external objects. Given the robot-obstacle distance vectors, we generate suitable escape trajectories for the robot using repulsive potential fields.

### 5.1.2 Related Work: Collision Detection and Avoidance

In this section, we briefly describe the related work regarding the topic of collision detection and avoidance based on optical sensors. The state of the art regarding automatic derivation of whole-body behaviors can be found in Section 3.2.

#### Robot-Obstacle Distance Computation

Since real-time collision avoidance is of fundamental importance in robotics, the topic has been widely studied and many different approaches have been developed throughout the years. An interesting work in this regard has been presented by Flacco et al. [Fla+12]. The presented approach computes distances between obstacles and the robot directly in a depth image provided by a 3D camera, which is significantly faster than other robot-obstacle distance computation methods. Later, the authors extend their approach to an integrated framework for collision detection, reaction and avoidance [DF12]. In contrast, we compute robot-obstacle distances in 3D space using point clouds and a 3D collision model of the robot, which is slower. However, our approach allows us to integrate multiple different sensors that supply point cloud data, e.g., RGB-D cameras and laser scanners. Furthermore, using a WBC-based approach, we can integrate additional tasks like joint limit avoidance. Another approach that applies convex hulls to represent robots and obstacles has been recently

presented by Han et al. [Han+18]. However, the approach focuses on motion planning and not on reactive control.

The approach for collision distance evaluation we use here is based on the KCCD library [TBF11; TF12]. KCCD uses a computationally efficient convex hull representation for rigid bodies. The original approach is intended to be used for self-collision avoidance of complex robots with e.g., tree structure. Here, we provide an extension to KCCD to include external objects observed by sensor data, which is required for real-time robot-obstacle collision avoidance.

## Collision Avoidance Control

Planning and control of collision avoidance behaviors is also a long-studied topic in robotics. A lot of methods are based on the concept of artificial potential fields [Kha85]. The work presented by Brock and Khatib [BK02], combines a method for real-time path modification and task-consistent robot control based on repulsive potential fields. While the approach generates task consistent avoidance motions, it is based on 2D laser scanner information. In contrast, the approach we present here uses 3D point cloud data, which allows more complex collision avoidance scenarios, e.g., between a human and robot sharing the same workspace or performing shared assembly. Also, we extend the concept of WBC with black-box optimization techniques in order to automatically discover soft priorities and other controller parameters, which are often difficult to obtain manually.

### 5.1.3 Robot-Obstacle Distance Computation

In this section we provide a brief overview of our approach for real-time robot-obstacle distance estimation. For a detailed description and evaluation, please refer to the related publication [Mro+20].

The approach for robot-obstacle distance computation is based on 3D point clouds, which can be obtained from different sensor sources. Here, we use RGB-D cameras as input and we perform the following preprocessing steps:

- i. Subtract the background from the depth image
- ii. Perform robot self-filtering in the depth image
- iii. Convert the raw depth data from multiple 3D cameras into a single point cloud
- iv. Cluster the remaining points using spatial information
- v. Convert the point cloud clusters into convex hulls

Figure 5.1 shows an overview on the processing steps, while a detailed description can be found in the related publication by Gea Fernández et al. [Gea+17a]. After preprocessing steps (i)-(iv), we have several 3D point clusters, which describe the external objects in the vicinity of the robot (see Figure 5.1(c)). Each cluster may contain a couple of hundred to a couple of thousand 3D points, depending on the size of the object and the resolution of the camera.

KCCD is a C++ library, which may evaluate distances between rigid bodies in real-time. Each body is modeled as a convex volume, which is described by a finite number of support points and a radius  $r$ . With this representation, arbitrary objects can be described, where the accuracy of the hull depends on the number of support points used. For example, a sphere



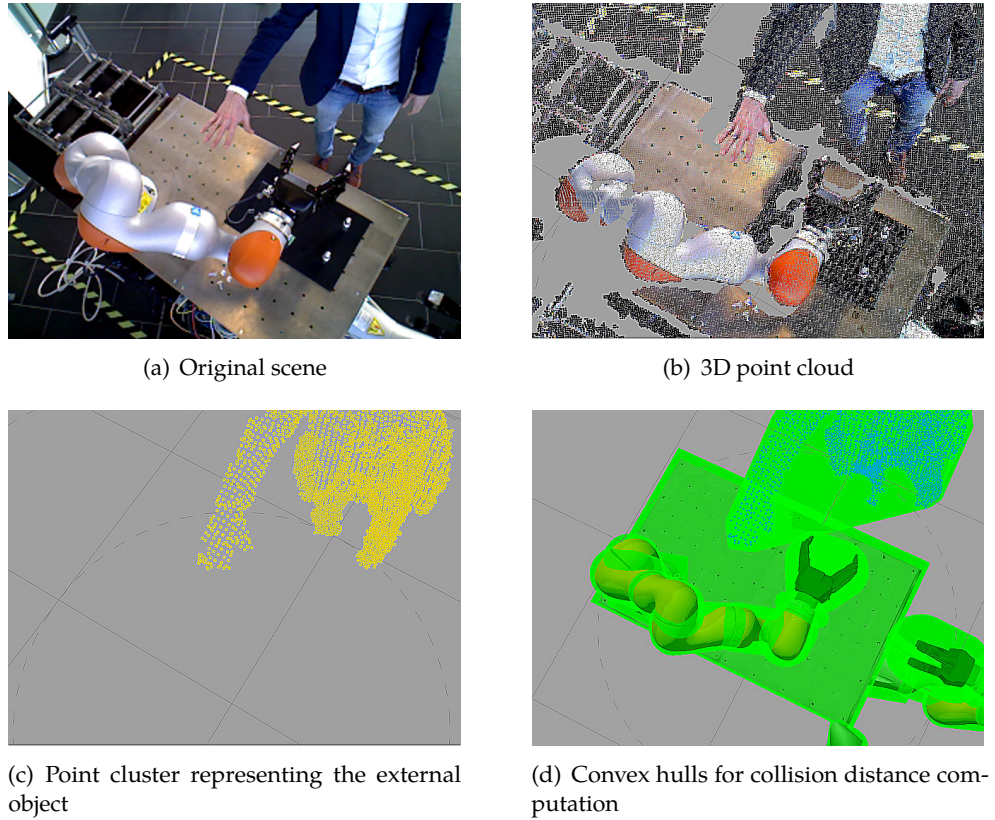


Figure 5.1: Overview of the most important sensor processing steps.

can be described by with a single point and a radius vector. More complex objects like e.g., a robot gripper may require many points and there is a need for trading off computational complexity and approximation accuracy. KCCD requires that the user models all robots and other collision bodies. Thus, it cannot deal with dynamic, unknown objects entering the workspace of the robot. To overcome this limitation, we extend the KCCD library to allow insertion of collision bodies into the model at runtime. To achieve this we provide a method to efficiently convert point clouds into KCCD collision bodies. The result of this conversion can be seen in Figure 5.1(a). Here, the raw point cloud cluster shown in Figure 5.1(c) is converted into a KCCD volume using an online algorithm. A detailed description of this algorithm can be found in the related publication by Mronga et al. [Mronga+20]. Using this approach, KCCD is able to evaluate robot-obstacle distances using the collision model of the robot and point cloud clusters, which are wrapped by KCCD volumes. The robot obstacle distance vectors computed by KCCD can be used to generate suitable avoidance motions, which is described in the following section.

#### 5.1.4 Task-Compliant Collision Avoidance using WBC

For the integration of avoidance control with other robot tasks like positioning in Cartesian space, we use the WBC approach described by (4.1). Cartesian pose constraints are enforced using (4.2). For avoidance motions we use repulsive potential fields, which provide a control

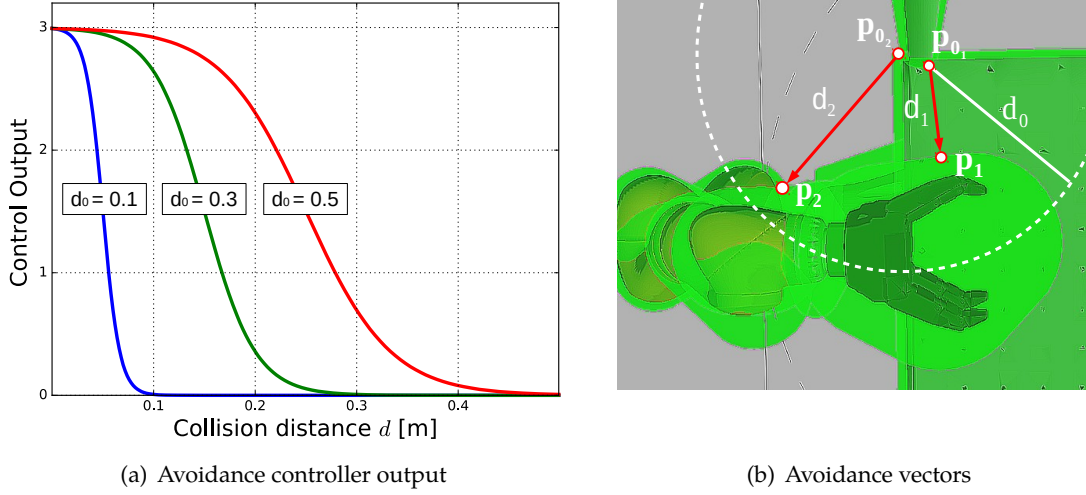


Figure 5.2: Generation of avoidance motions using KCCD and potential fields.

output of  $\mathbf{v}_d = (\mathbf{v}_t, \mathbf{0}^{3 \times 1})^T$  with

$$\mathbf{v}_t = \begin{cases} \mathbf{K}_p \frac{\mathbf{p} - \mathbf{p}_0}{d} S(d), & \text{if } d < d_0 \\ \mathbf{0}, & \text{else} \end{cases} \quad \mathbf{v}_t \leq \bar{\mathbf{v}}_t \quad (5.1)$$

where  $\mathbf{p}, \mathbf{p}_0 \in \mathbb{R}^3$  are the actual robot position and the position of the potential field center,  $d_0$  the maximum influence distance of the potential field,  $\bar{\mathbf{v}}_t$  the maximum controller output or saturation term and  $\mathbf{K}_p \in \mathbb{R}^{3 \times 3}$  is a diagonal matrix containing the 3 feedback gain constants.  $S(d)$  is a sigmoid function of the distance  $d = \|\mathbf{p} - \mathbf{p}_0\|_2$  between the robot and the potential field center:

$$S(d) = \left( 1 + e^{\alpha \left( 1 - 2 \frac{d_0 - d}{d_0} \right)} \right)^{-1} \quad (5.2)$$

The factor  $\alpha = 6$  is chosen empirically. If the robot is close to an obstacle, the collision distance becomes small and the repulsive control action will be maximal, as  $S(d) \approx 1$ . If the collision distance  $d$  is close to the maximum influence distance  $d_0$  of the potential field, the exponential term becomes large and  $S(d) \approx 0$ . Thus, for  $d \approx d_0$ , the control output  $\mathbf{v}_d$  vanishes. Figure 5.2(a) illustrates the development of the control output with respect to the computed collision distance for  $k_p = 3$  and different  $d_0$ . The Sigmoid function implements a smooth transition if the robot enters the influence sphere of a potential field (in contrast to e.g., piecewise linear functions). Sigmoid functions are finite, which ensures a finite repulsive velocity and a more stable robot behavior compared to having a reciprocal (or even squared reciprocal) dependency on the robot-obstacle distance.

To allow collision avoidance with the entire robot body, we define one avoidance controller per robot link. The potential field center  $\mathbf{p}_{0,i}$  and the corresponding control point  $\mathbf{p}_i$  for the  $i$ -th robot link are assigned to the two closest points on the collision volumes of obstacle and robot link, respectively. This procedure is illustrated for the last two links of a KUKA iiwa

robot in Figure 5.2(b). Using (5.1) we convert the safety distances  $d_i$  into equality constraints of the form  $\mathbf{J}_i \dot{\mathbf{q}} = \mathbf{v}_{d,i}$ , where  $\mathbf{v}_{d,i}$  is computed according to (5.1) and  $\mathbf{J}_i$  is the Jacobian associated with robot link  $i$ . We integrate the collision avoidance tasks with other tasks like Cartesian position control into the instantaneous optimization problem in (4.1). As in Section 4.3 we use a standard QP solver [FBD08] for computing the optimal solution of this problem. The problem is obviously over-constrained, which means that an accurate solution does not exist, especially if multiple collision avoidance tasks are simultaneously active. To avoid unnecessary restriction of robot motion in free space, the soft task priorities for the collision avoidance tasks are adapted with respect to the collision distance as follows:

$$w_i = \begin{cases} (d_0 - d)/d_0, & d < d_0 \\ 0, & \text{else} \end{cases} \quad \forall i \quad (5.3)$$

The weights are zero if the robot is moving far from any obstacle, so that the avoidance behaviors do not interfere with the main task. When a robot link approaches an obstacle, the weights gradually increase, implementing a smooth transition between unconstrained and constrained robot motion. This way we can smoothly integrate avoidance motions with other simultaneously running tasks.

The solution described here is governed by several parameters, namely control gains  $\mathbf{k}_p$ , saturation terms  $\bar{\mathbf{v}}$ , maximum influence distance  $d_0$  of the potential field controllers, as well as the soft task priorities  $\mathbf{w}$ . Manual choosing and fine-tuning these parameters can be time-consuming and may lead to a suboptimal solution. To overcome these issues, we introduce an automatized procedure based on black-box optimization for deriving these task parameters. The approach is described in the following section.

### 5.1.5 Optimization of WBC Parameters

We propose to apply black-box optimization<sup>1</sup> based on evolutionary techniques to generate an optimal parameter set for WBC. The advantage of black-box optimization in general is that we do not require detailed knowledge about the fitness landscape. However, the design of the cost or fitness function requires particular care. Another disadvantage is that, depending on the optimization method, many evaluations of the fitness function may have to be performed. Thus, we rely on a simulation of the scenario for fitness evaluation.

When combining position control with collision avoidance using WBC, the given tasks have conflicting objectives, i.e., avoidance behaviors might prevent reaching the reference position of the manipulator. In WBC with conflicting objectives, trading off the performance of the individual task constraints is required, a process that is typically done by manually tuning of task priorities and other control parameters. For optimization of the parameters, a global fitness function describing the overall robot performance is required. Here, we propose the

<sup>1</sup> Note that we refer to offline optimization of the control parameters here, not to be confused with the instantaneous optimization problem as in (4.1)



following fitness function:

$$f = \lambda \left( \underbrace{\sum_i \alpha_i f_{cd}^i}_{\text{Collision Fitness}} + \underbrace{\sum_j \frac{\alpha_j}{3} (f_{ss}^j + f_{po}^j + f_{st}^j)}_{\text{Position Fitness}} \right) \quad (5.4)$$

This fitness function sums up the fitness of all task constraints, divided into collision and position fitness. The  $\alpha$ -coefficients are weighting factors used to balance the contribution of each individual task constraint. The individual fitness measures are computed as follows:

$$\begin{aligned} \text{Steady State Error} \quad f_{ss} &= S(e(t_{ss})) \\ \text{Percentage Overshoot} \quad f_{po} &= S(\|\mathbf{p}_{max} - \mathbf{p}_{min}\|_2) \\ \text{Settling Time} \quad f_{st} &= S(t_{ss} - t_0) \\ \text{Min. Col. Distance} \quad f_{cd} &= S(d_{min}) \end{aligned} \quad (5.5)$$

Here  $e = \|\mathbf{p}_r - \mathbf{p}\|_2$  is the position error<sup>2</sup>,  $t_0, t_{ss}$  the start time of the motion and the settling time of the controller,  $\mathbf{p}_{max}, \mathbf{p}_{min}$  are the maximum and minimum position observed during transient behavior and  $d_{min}$  the minimum distance to a collision object observed during motion. The term  $S$  is again a Sigmoid function, which normalizes the fitness measures to the interval  $[0, 1]$ . It also inverts the slope of the functions  $f_{ss}, f_{po}, f_{st}$  as they are supposed to be maximized here. The regularization term  $\lambda$  is computed as follows:

$$\lambda = e^{-\frac{1 - \min(f^i)}{\min(f^i)}} \quad (5.6)$$

The term punishes results with low fitness values of individual tasks. This way, solutions that represent a trade-off between different tasks will be favored compared to solutions where one task is fully achieved while another task is ignored.

### 5.1.6 Results

For evaluation of our approach, we use the iMRK robot described in Section 1.2. The robot control software runs on an industrial PC with Intel Core i7 4790K 4 x 4.00 GHz, while the sensor processing part runs on a standard desktop PC with Intel Core i7-2600 CPU 4 x 3.40 GHz. The iMRK system is described in more detail by Gea Fernández et al. [Gea+17b]. The evaluation of the real-time robot-obstacle distance computation method mentioned in Section 5.1.3 can be found in the work by Mronga et al. [Mro+20]. Here, we focus here on the WBC-related aspects and the parameter optimization part.

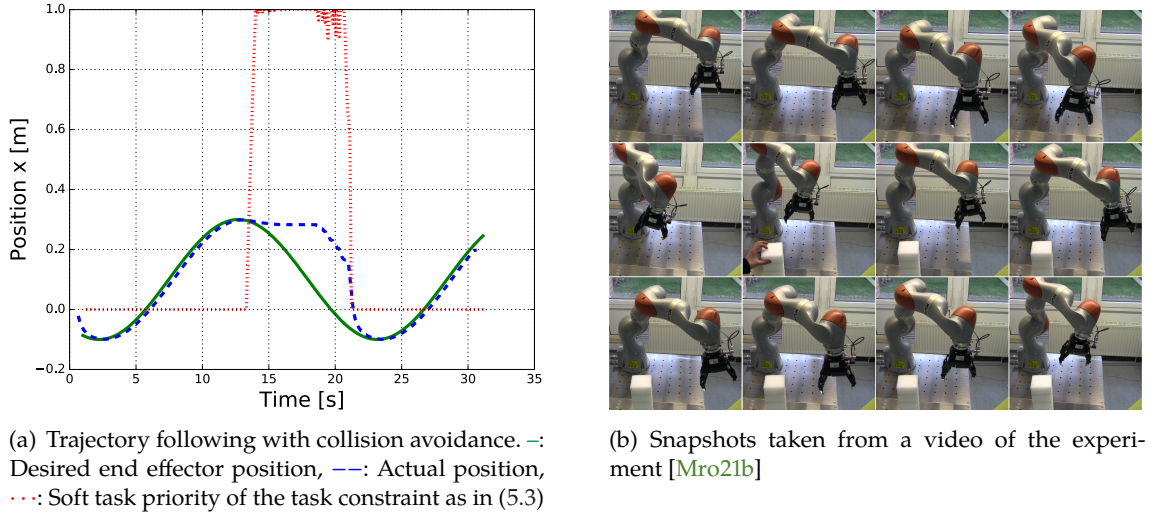


Figure 5.3: Results on task-compliant collision avoidance using WBC.

### Task-Compliant Collision Avoidance using WBC

We evaluate the general functionality of our WBC approach for generating suitable escape trajectories for collision avoidance. For evaluation, we use a single arm of the iMRK system as shown in Figure 1.3 and mount it vertically on the top plate of a table. The arm is supposed to follow a circular end effector trajectory, while avoiding an obstacle, which is randomly placed in the robot workspace. For obstacle detection, we use a single ASUS Xtion RGB-D camera. The WBC control parameters are chosen as follows. For Cartesian position control as in (4.2) we set the feedback gain and saturation values to  $k_{p,i} = 1.5$ ,  $\bar{v}_i = 0.5$ ,  $\forall i$ . The task weights used in the optimization problem described by (4.1) are set to fixed values  $w_i = 1$ ,  $\forall i$ . For the avoidance controller as described by (5.1) we set the control gain and saturation to  $k_{p,i} = 0.05$ ,  $\bar{v}_i = 0.5$ ,  $\forall i$  and the influence distance to  $d_0 = 0.5\text{m}$ . The weights of the collision avoidance task are computed automatically according to (5.3). The results are illustrated in Figure 5.3. The trajectory is accurately followed until the end effector enters the influence zone of the obstacle. Here, the task priority for collision avoidance is increased and the robot deviates from its reference path. The figure also shows that the transient behavior between free and constrained motion is not satisfactory. Having chosen too aggressive control gains, we observe repeated activation and deactivation of the collision avoidance task in the transient zone. Although each task constraint performs well on its own, the combination of position control and obstacle avoidance leads to suboptimal overall behavior in this case. As discussed in section 5.1.1, manual parametrization of whole-body controllers can be time-consuming and, as in this case, may lead to suboptimal results. In the next section, we evaluate our approach for automatic derivation of WBC task parameters, which is intended to overcome these problems.

### Black-Box Optimization of WBC Parameters

We apply a genetic algorithm (GA) from the DEAP evolutionary computation framework [For+12] for black-box optimization of WBC parameters. This choice is motivated

<sup>2</sup> For the sake of readability we omit the orientation error here.

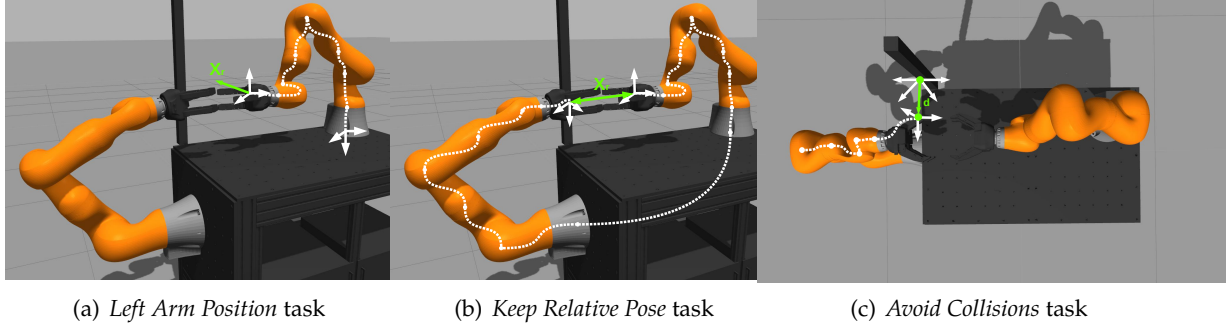


Figure 5.4: Illustration of tasks used in the experimental setup. The dotted white lines indicate which parts of the robot are used for a task.

by the fact that the fitness function as defined in (5.4) may comprise multiple extrema due to conflicting objectives. Finding the global maximum in such a fitness landscape can be difficult and satisfactory results can only be achieved by using derivative-free optimization methods like GA. Given that GAs typically need many fitness evaluations, we simulate the robot using Gazebo [Fou21c] as a simulation environment. We program a whole-body controller comprising three robot tasks, which are framed as task constraints in the optimization problem (4.1). Figure 1reffig:subtasks illustrates the tasks. First, we apply a position controller as in (4.2) to follow an end effector trajectory with the left arm (*Left Arm Position* task). Secondly, the end effector of the right arm shall maintain a fixed pose with respect to the end effector of the left arm (*Keep Relative Pose* task). Third, the robot is supposed to avoid collisions with obstacles (*Avoid Collisions* task)<sup>3</sup>. The only obstacle is given by the vertical bar, which is in the path of the right gripper. The position of the bar is fixed and known in this case, since we do not want perception errors to influence the results. The experimental setup includes many commonly encountered problems when specifying tasks for WBC, like conflicting constraints, dual arm coordination and redundant dof. In GAs, the candidate solutions for the parameters to be optimized are encoded as so-called individuals, which are iteratively refined using the biologically inspired principles of mutation, crossover and selection [SGK05]. Here, each individual encoded in the GA comprises the control parameters of the three tasks, including the proportional control gains  $k_p$ , the task weights  $w$ , the saturation terms  $\bar{v}$ , as well as the maximum influence distance  $d_0$  of the potential field controller. In summary each individual comprises the following parameters:

$$\begin{Bmatrix} k_p^L & w^L & \bar{v}^L \\ k_p^K & w^K & \bar{v}^K \\ k_p^A & d_0^A & \bar{v}^A \end{Bmatrix} \quad (5.7)$$

The superscripts correspond to the respective tasks *Left Arm Position*, *Keep Relative Pose* and *Avoid Collisions*. Note that the task weights of the avoidance controller are computed automatically according to (5.3), thus they will only be optimized indirectly through the influence distance  $d_0$ . Since Cartesian positioning and avoidance tasks have 6 and 3 dof,

<sup>3</sup> We only consider collisions of the right gripper here. However, the solution can be extended to include all links of the robot.

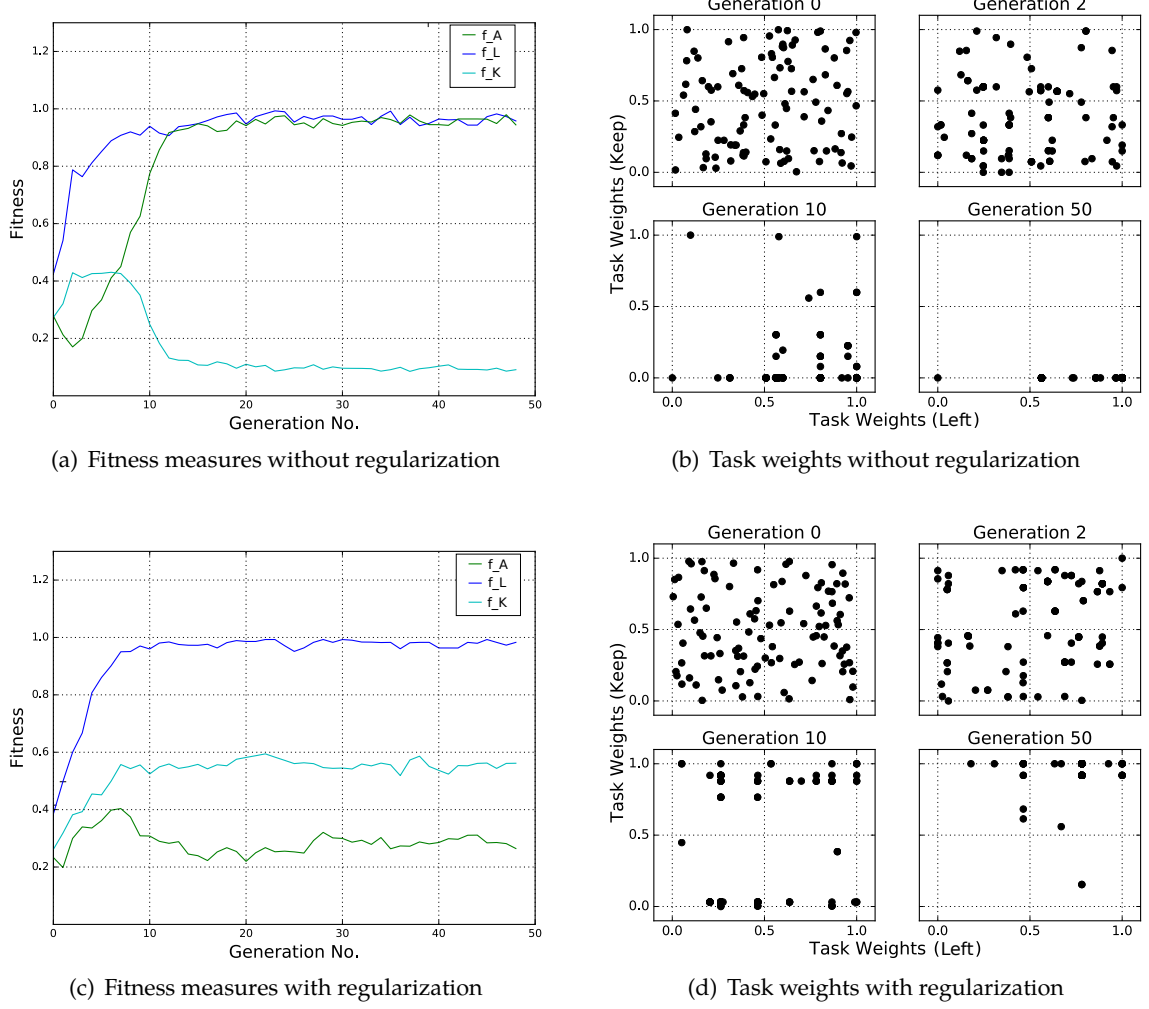


Figure 5.5: GA results. Development of individual fitness measures and the global fitness (left). Development of task weights of the *Keep Relative Pose* and the *Left Arm Position* task (right)

respectively, we obtain  $6 \cdot 3 + 6 \cdot 3 + 3 \cdot 3 = 45$  free parameters in total. However, for simplification we set the parameters of all dof of a task to the same value, which reduces the number of free parameters to 9. We initially use a population of 100 individuals, drawn randomly from the feasible parameter space. The fitness function is chosen according to (5.4), (5.5) and (5.6) as follows:

$$f = \lambda \left( \underbrace{\alpha_A f_{cd}^A}_{f^A} + \underbrace{\frac{\alpha_L}{3} (f_{ss}^L + f_{po}^L + f_{st}^L)}_{f^L} + \underbrace{\frac{\alpha_K}{3} (f_{ss}^K + f_{po}^K + f_{st}^K)}_{f^K} \right) \quad (5.8)$$

where  $\lambda$  is the regularization term and  $f^i$ ,  $i \in \{A, L, K\}$  the fitness measures for the individual tasks (*Avoid*, *Left*, *Keep*). We select the weighting coefficients as  $\alpha^A = 0.5$ ,  $\alpha^L = 0.2$  and  $\alpha^K = 0.3$ .

Figure 5.5 shows the development of the fitness function and the evolution of the individuals

Subtask	Optimized Parameter Set			
Step	$k_p^L = 2.651$	$w^L = 1.000$	$\bar{v}^L = 0.361$	
Keep	$k_p^K = 1.578$	$w^K = 0.919$	$\bar{v}^L = 0.371$	
Avoid	$k_p^A = \mathbf{0.001}$	$d_0^A = 0.654$	$\bar{v}^A = 0.768$	
Subtask	Manually Tuned Parameter Set			
Step	$k_p^L = 1.500$	$w^L = 1.000$	$\bar{v}^L = 0.200$	
Keep	$k_p^K = 2.500$	$w^K = 1.000$	$\bar{v}^L = 0.200$	
Avoid	$k_p^A = \mathbf{0.100}$	$d_0^A = 0.500$	$\bar{v}^A = 0.500$	
Fitness	$f^L$	$f^K$	$f^A$	$f$
Optimized	<b>0.956</b>	0.600	0.611	0.347
Manual	<b>0.618</b>	0.600	0.615	0.314

Table 5.1: Optimization results: Optimized and manually tuned parameter set. Individual fitness measures  $f^L$ ,  $f^K$ ,  $f^A$  and global fitness  $f$  for the respective parameter sets

for two different cases: With regularization term (bottom row) and without regularization term ( $\lambda = 1$ , top row). Given the conflict of position and avoidance control, optimal fulfillment of all tasks is not possible. As a result, the GA may favor results where one task is fully achieved, while the performance of another task becomes small. This problem can be observed in Figure 5.5(a). Here, the regularization term is omitted. As a result, the GA converges to a solution, where the task weights of the *Keep Relative Pose* task tend towards zero (see Figure 5.5(b)). Consequently, the right arm does not follow the left arm at all, which leads to high fitness for the *Avoid Collisions* and *Left Arm Position* tasks, small fitness for the *Keep Relative Pose* task and an average overall fitness. The introduction of the regularization term in (5.4) avoids such problems by punishing results with low fitness values for the individual tasks. Figure 5.5(c) shows that a tradeoff is achieved and none of the individual fitness functions converges to extremely small values. Also, the task weights of the *Keep Relative Pose* task assume average values now. The resulting optimized parameter set is shown in Table 5.1. We compare the optimized parameters with a parameter set that has been found by manually tuning the individual tasks one at a time. The most prominent difference is the proportional gain value of the *Avoid Collisions* task, which ends up with a low value after black-box optimization. The resulting overall robot behavior is quite different, especially when entering the influence distance of an obstacle. Figure 5.6(a) shows the step response of the left end effector in the proximity of the obstacle. Obviously, the proportional gains for the manually selected parameter set have been chosen too high for this specific scenario, which results in a suboptimal transient behavior. In our case, the parameter optimization strategy automatically avoids this problem by minimizing the overshoot and settling time and thus proposing a lower proportional gain for the potential field controller. The fitness values for the individual tasks and the global fitness for both parameter sets are also shown in Table 5.1. As it can be seen, the *Keep Relative Pose* and *Avoid Collisions* tasks perform similarly for both parameter sets. The *Left Arm Position* task on the other hand performs much worse for the manually tuned parameter set, which is due to the long settling time of the controller, as can also be seen in Figure 5.6(a). Having found an optimal set of WBC parameters in an automated manner like this, we evaluate its quality in a human-robot coexistence scenario in the following section.

### Human-Robot Coexistence Scenario

We transfer the optimized WBC parameters found in simulation to the real robot system and evaluate it in a human-robot coexistence scenario. The left robot arm is supposed to follow a continuous trajectory, while a human moves into the path of the robot. Again, the right arm end effector should maintain the relative pose to the left arm end effector. Figure 5.7 shows video snapshots of the human-robot collision avoidance. Figure 5.6(b) shows the reference trajectory along with actual trajectory observed by using the optimized and the manually tuned parameter set. As before, the optimized parameter set provides a smoother transient behavior between free and constrained motion. Far away from any collision, the end effector trajectory is followed accurately. The results suggest that the approach is able to create optimal parameter sets for such kind of robot task in an automatized manner. In this case, the parameter set we found provides safe and jerk-free reactions in a human-robot coexistence scenario.

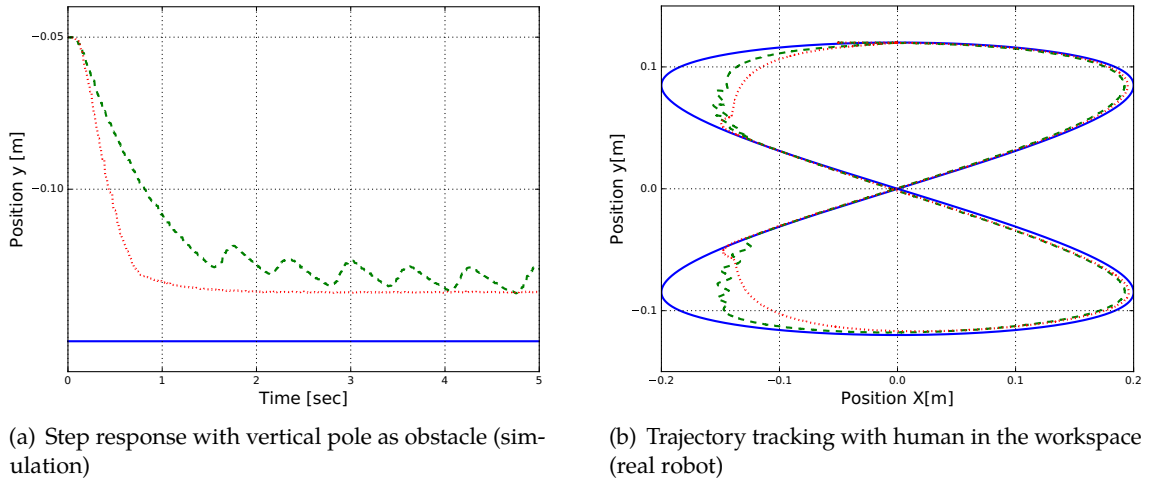


Figure 5.6: Comparing the robot behavior for the optimized and manually tuned WBC parameters. —: Reference trajectory, —: Trajectory with manually tuned parameters, ···: Trajectory with optimized parameters



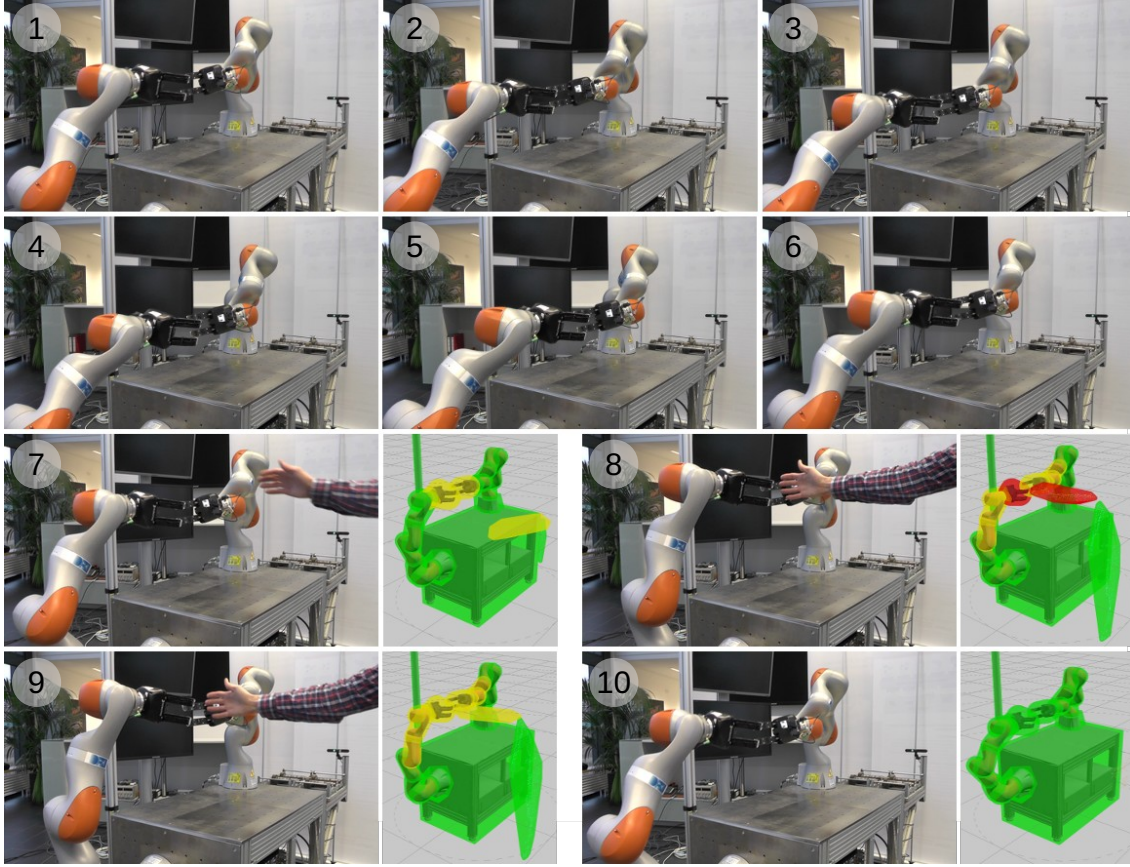


Figure 5.7: Trajectory tracking and collision avoidance with a human entering the workspace of the robot, screenshots from video [Mro21a].

## 5.2 Optimization of Soft Task Priority Functions

We extend the approach for optimizing WBC task parameters as described in the previous section. While the former method optimizes fixed task parameters like control gains or soft task priorities, we consider optimization of task priorities as function of time here. This provides higher flexibility for complex tasks with time-varying objectives.

### 5.2.1 Motivation

When dealing with multiple, conflicting tasks in WBC, the choice of the prioritization scheme is of utmost importance. For complex robots like humanoids, the number of tasks can easily exceed a dozen, in addition to multiple constraints that must be considered simultaneously. Strict prioritization schemes are not well suited in such scenarios. On the one hand they often unnecessarily constrain the available robot dof and high-priority tasks may block tasks with lower priority. On the other hand, the task hierarchy must be selected in advance, and online switching of the task hierarchy may result in discontinuities in the control law [MKK09]. Soft task priorities are more promising here, as they allow smooth transitions between tasks, which facilitates execution of task sequences. Furthermore, soft priorities can have temporal profiles, which may optimally exploit the robot capabilities throughout the execution of a task. On the downside, tuning temporal profiles of soft task priorities is a difficult, time-consuming

task and may result in suboptimal robot behavior, e.g., in terms of controller stability and task consistency. Consequently, there is a need to develop automatized methods for derivation of soft task priority functions in WBC.

In this section, we present an approach that utilizes machine learning methods to automatically optimize soft task priorities as a function of time. We start from an initial guess of the soft task priorities obtained from PbD as described in Chapter 4. The soft task priorities derived from user demonstrations might be suboptimal for several reasons. First, the user demonstrations may not capture the structure of the task and the most important task constraints. Secondly, when executing the learned tasks on the target robot, the behavior might be different from what is expected because of sensor uncertainty or not correctly modeled constraints. Finally, when executing task sequences, task transitions can be smoothly performed by blending task priority functions, which is hard to model by hand though. In all three cases black-box optimization of task priorities can be performed to improve the overall robot behavior.

In fact, machine learning can be applied with the goal of improving several aspects of the overall robot performance in WBC, like task consistency [Sil+19; LPS16; Mod+16b], accuracy [DRS15], ergonomics [Bus+17], safety [Mod+16a], smoothness, manipulability or dof usage. The application of black-box optimization assumes that a suitable cost function has been defined by the user. Several functions have been proposed in literature for this purpose. The approaches presented by Modugno et al. [Mod+16b; Mod+16a] utilize a combination of cumulative distance and global control effort as cost. Charbonneau et al. [Cha+18] compare three criteria applied in a static walking task on a humanoid robot. The first, similarly to the work presented by Modugno et al. [Mod+16b; Mod+16a], penalizes Cartesian task error and control effort, the second focuses on robustness of the ZMP controller, the third is a combination of both. Lober et al. [LPS16] propose a combination of position tracking error, goal error and kinetic energy cost, even though their focus is on optimization of task trajectories and not on soft task priorities. Dehio et al. [DRS15] apply a combination of tracking error and energy cost. Here, the latter is implemented by minimizing the sum of control torques generated by each individual task. This way, irrelevant tasks are filtered out by automatically reducing their respective soft task priorities.

The selection of a global cost function for optimization requires a certain amount of expertise and may be just as difficult and time-consuming as manual tuning of task priorities. The quality of the obtained solution depends strongly on the choice of this function. To investigate the effect of different cost functions, we perform an empirical comparison using functions that can be found in literature for the optimization of soft task priorities.

### 5.2.2 Approach

In this section, we apply an optimization-based WBC approach on velocity level. Each task is described as part of the cost function of an unconstrained QP:

$$\min_{\dot{\mathbf{q}}} \quad \left\| \sum_{i=1}^P w_i (\mathbf{J}_i \dot{\mathbf{q}} - \mathbf{v}_{d,i}) \right\|_2 \quad (5.9)$$

We employ a one-dimensional soft task priority  $w$  per task in order to simplify the optimization problem. In contrast, the approach in (2.15), employs one soft task priority per task space



variable. We generate the control actions  $\mathbf{v}_{d,i}$  by the means of a Cartesian position controller as in (2.17). The soft task priorities are functions of time  $w = w(t)$ . We model their temporal profile using a weighted combination of radial basis functions (RBFs):

$$w(t, \boldsymbol{\theta}) = \frac{\sum_i \alpha_i \Psi_i}{\sum_i \Psi_i} \quad (5.10)$$

where each RBF is represented by a Gaussian:

$$\Psi_i(t, \mu, \sigma) = e^{-\frac{1}{2} \left( \frac{t-\mu}{\sigma} \right)^2} \quad (5.11)$$

The shape of each soft task priority profile is governed by a parameter set  $\boldsymbol{\theta} = \{\alpha_i, \mu_i, \sigma_i\}_{i=1}^K$  comprising the weights  $\alpha_i$ , means  $\mu_i$  and variances  $\sigma_i$  of the Gaussians. This parameter set is learned using black-box optimization given a suitable cost function. Thereby, the soft task priorities obtained from user demonstrations as described in Section (4.5.3) can be used as initial guess to speed up optimization. The choice of a suitable cost function for optimization depends on the application. When combining different objectives to one global cost function, the global cost may be sensitive to proper scaling or weighting of the individual cost functions. To highlight these issues, we use a combination of different optimization criteria:

$$\begin{aligned} \text{Tracking Error} \quad f &= \frac{1}{T} \sum_{t=t_0}^{t_e} \|\mathbf{e}_t\|_2 \\ \text{Goal Error} \quad f &= \|\mathbf{e}_{t_e}\|_2 \\ \text{Kinetic Energy} \quad f &= \frac{1}{T} \sum_{t=0}^T \|\mathbf{v}_t\|_2^2 \\ \text{Angular Jerk} \quad f &= \|\ddot{\mathbf{q}}_{\max}\|_2 \end{aligned} \quad (5.12)$$

Here,  $\mathbf{e}$  is the pose error,  $T$  is the number of time steps between start time  $t_0$  and end time  $t_e$  of the task,  $\mathbf{v}$  the spatial velocity of the robot end effector and  $\ddot{\mathbf{q}}_{\max}$  the maximum jerk (change of acceleration) in joint space. Most of these criteria have been used in literature before, in particular Ivaldi et al. [Iva+12] give a good overview on optimality criteria in human motor control. Here, we attempt to optimize the soft task priorities for a multi-objective control problem. Thus, the landscapes of the cost functions will be multi-modal. For example, when combining position control and obstacle avoidance as illustrated in Section 5.1, optimizing the soft task priorities will be prone to local optima. To deal with this issue, we employ the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hot01] for black-box optimization, which is a stochastic, derivative-free approach that belongs to the class of evolutionary algorithms. It is particularly well suited for complex, multi-modal cost functions.

### 5.2.3 Results

In this section, we evaluate the optimization of soft task priorities for WBC using CMA-ES as black-box optimizer. We compare the effect of using different cost functions. We apply the criteria defined in the previous section and weighted combinations of them. As experimental system, we use the dual-arm KUKA iiwa robot (see Figure 1.3) and apply the trajectories obtained from the *Rotate Object* data set as illustrated in Section 4.5.3. In this data set, we

observe three different tasks: *Right Arm Motion* (R), *Left Arm Motion* (L) and *Keep Relative Pose* (K). As the tasks are 6-dof Cartesian tasks and the robot has 14 dof, the problem is over-constrained. By tuning the soft task priority functions, a trade-off between the different control objectives can be obtained. We define one soft task priority function as defined in (5.10) per task. The parameters of the task priority functions  $\{\theta_L, \theta_R, \theta_K\}$  comprise the weights, means and variances of the Gaussian RBFs as in (5.11). We choose  $K = 10$  RBFs per task priority, thus we obtain 90 optimization parameters. As CMA-ES belongs to the class of evolutionary algorithms, the optimization parameters are denoted as individuals. In CMA-ES, new candidate solutions are generated from a population of individuals using stochastic sampling. In each iteration of the algorithm, which is referred to as generation, a cost function is evaluated for each individual. In each generation, only the most promising candidates are kept. In the next generation a new population is generated by sampling from a multi-variate normal distribution. Mean and variance of the distribution evolve over the generations. Apart from the cost function, only a small number of hyper-parameters must be tuned in CMA-ES. The most important ones are the initial variance  $\sigma_0$ , which is related to the exploration rate, and the number of individuals  $N_i$  per generation. As we normalize the RBF parameters to zero mean and unit variance before feeding them in the optimizer, we can choose  $\sigma_0 = 1$ . The number of individuals per generation is set to  $N_i = 100$ . We perform  $N_g = 300$  iterations of CMA-ES, which gives us a total of 90000 evaluations of the cost function. As it is not possible to perform so many evaluations on the real robot, we rely on a simulation of the system. As a simulator we use the RaiSim physics engine [HLH18]. We select the following cost functions for comparison:

$$f_1 = \frac{1}{2} \left( \underbrace{\frac{1}{T} \sum_{t=t_0}^{t_e} \|\mathbf{e}_t^K\|_2}_{\text{Tracking Cost (Keep)}} + \underbrace{\frac{1}{T} \sum_{t=t_0}^{t_e} \|\mathbf{v}_t\|_2^2}_{\text{Kinetic Energy (all)}} \right) \quad (5.13)$$

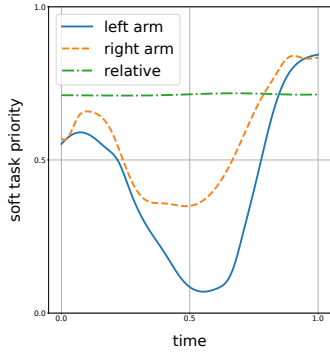
$$f_2 = \frac{1}{2} \left( \underbrace{\frac{1}{T} \sum_{t=t_0}^{t_e} \|\mathbf{e}_t^K\|_2}_{\text{Tracking Cost (Keep)}} + \underbrace{\|\mathbf{e}_{t_e}^L\|_2}_{\text{Goal Cost (Left)}} \right) \quad (5.14)$$

$$f_3 = \frac{1}{3} \left( \underbrace{\frac{1}{T} \sum_{t=t_0}^{t_e} \|\mathbf{e}_t^K\|_2}_{\text{Tracking Cost (Keep)}} + \underbrace{\|\mathbf{e}_{t_e}^L\|_2}_{\text{Goal Cost (Left)}} + \underbrace{\frac{1}{T} \sum_{t=t_0}^{t_e} \|\mathbf{v}_t\|_2^2}_{\text{Kinetic Energy (all)}} \right) \quad (5.15)$$

$$f_4 = \frac{1}{3} \left( \underbrace{\frac{1}{T} \sum_{t=t_0}^{t_e} \|\mathbf{e}_t^K\|_2}_{\text{Tracking Cost (Keep)}} + \underbrace{\|\mathbf{e}_{t_e}^L\|_2}_{\text{Goal Cost (Left)}} + \underbrace{\|\ddot{\mathbf{q}}_{max}\|_2}_{\text{Smoothness Cost}} \right) \quad (5.16)$$

The optimization results are illustrated in the Figures 5.8 and 5.9, as well as in Appendix E. Figure 5.8 shows the initial soft task priorities obtained from user demonstrations, as well as the resulting task priorities after optimization using the cost functions  $f_1, \dots, f_4$ .

In Figure 5.8(b) it can be seen that the priority of the *Keep Relative Pose* task constantly assumes a high value, while the priorities of the other tasks tend towards zero. The former effect is



(a) Initial task priorities obtained from PbD

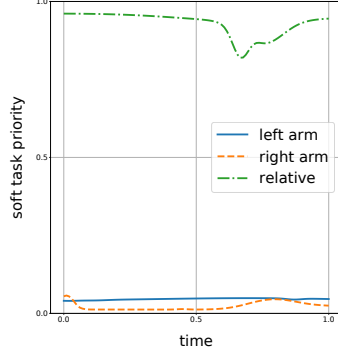
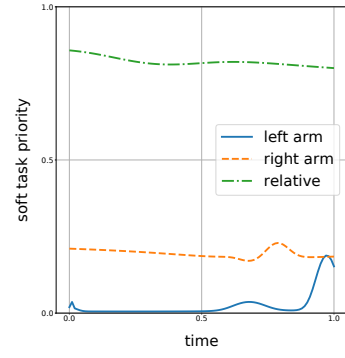
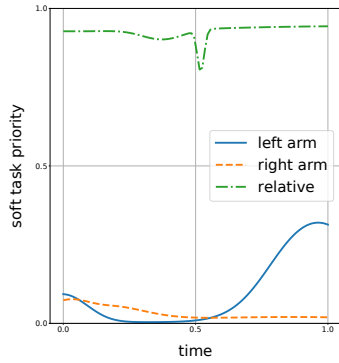
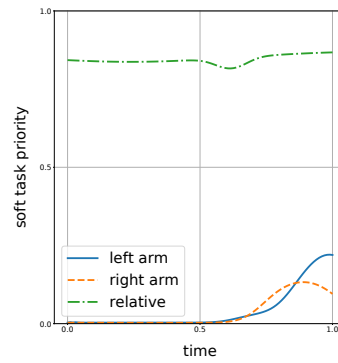
(b) Optimized task priorities, cost function  $f_1$ (c) Optimized task priorities, cost function  $f_2$ (d) Optimized task priorities, cost function  $f_3$ (e) Optimized task priorities, cost function  $f_4$ 

Figure 5.8: Resulting soft task priorities for the three tasks after optimization with CMA-ES

related to the first part of the cost function  $f_1$ , which minimizes the tracking error of the *Keep Relative Pose* task. The latter effect is resulting from the kinetic energy minimization of  $f_1$ , which drives the soft priorities of the less important tasks towards low values. This way, irrelevant tasks can be filtered out and the remaining dof of the robot can be utilized to perform additional tasks. The resulting trajectories shown in Figure E.1 in the appendix illustrate that the *Keep Relative Pose* task is tracked most accurately, while the tracking performance of the *Right Arm* and *Left Arm* tasks is degraded.

Figure 5.8(c) shows the optimized soft task priorities using cost function  $f_2$ , which includes the target pose error of the left arm as additional cost (goal cost) instead of the kinetic energy. The task priority for the *Left Arm* task increases towards the end of the task in order to comply with that demand. The task priority of the *Right Arm* task assumes medium values.

Cost function  $f_3$  is a combination of  $f_1$  and  $f_2$ . The resulting task priorities shown in Figure 5.8(d) reflect the demands of the given criteria. While the task priority of the *Keep Relative Pose* task is constantly high, the other task priorities assume low values. However, the *Left Arm* task priority increases towards the end of the task, which is due to the goal cost (left) included in the cost.

Finally, cost function  $f_4$  is a modification of  $f_3$ , which uses the maximum angular jerk as smoothness cost instead of the kinetic energy. The effect of using this cost function can be observed in Figure 5.8(e), which shows that the *Right Arm* task, as well as the *Left Arm* task

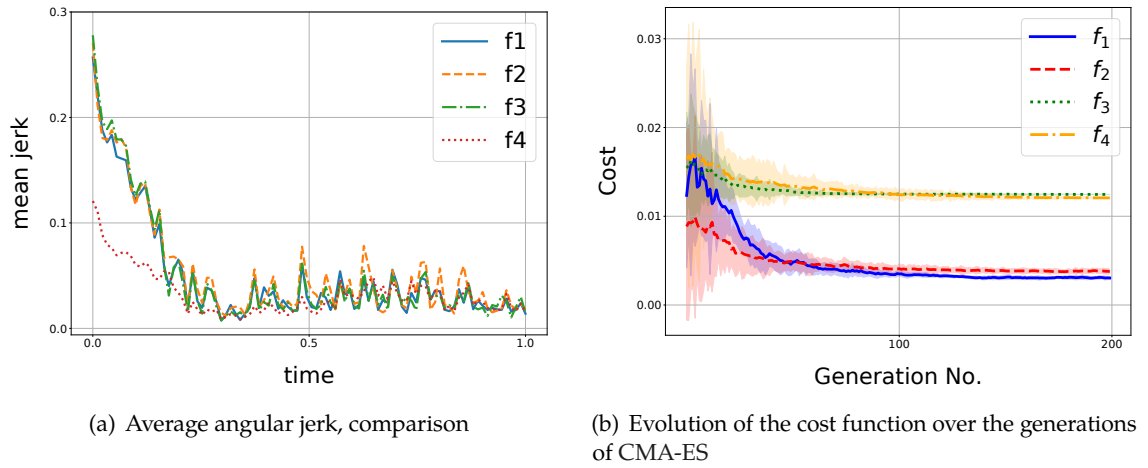


Figure 5.9: Result on task priority optimization.

assume zero task weights at the start of the motion and gradually increase towards the end of the task. We compare the average jerk over all joints using the different cost function in Figure 5.9(a). Especially in the beginning of the movement, using  $f_4$  as cost function produces significantly lower jerk.

### 5.3 Discussion

In this chapter we illustrate methods to optimize task priorities and other WBC parameters in order to improve whole-body behaviors once they are deployed on the target robot. Section 5.1 focuses on optimization of static WBC parameters, while Section 5.2 deals with task priority functions that have a temporal profile. While we evaluate the approaches on a fixed-base, industrial dual-arm robot, they are transferable to more complex systems like humanoids. However, the evaluation of the cost function usually relies on simulation, which becomes more complex and less accurate with increasing system complexity. Furthermore, not every aspect of a robot task can be easily modeled in simulation, especially when considering dexterous manipulation tasks or other complex environment interaction. Thus, the effort of creating a realistic simulation may exceed the effort for parameter tuning.

Still, the application of black-box optimization for task priority optimization in WBC is promising. Especially when dealing with temporal profiles of task priorities, the optimizer may produce a solution that a human expert cannot create by hand. When dealing with multiple, inconsistent tasks or task sequences, black-box optimization can produce more stable and smooth control signals, time-optimal task transitions and more consistent behaviors than manual tuning.

Naturally, the choice of the cost function has a major influence on the result of optimization and thus must be chosen with care. The criteria evaluated in Section 5.2 usually work well for certain scenarios. However, finding a generic criterion for a wide range of robot tasks is difficult.

# ARC-OPT: An Adaptive Whole-Body Control Framework

# 6

In this chapter we describe a modular and adaptive WBC framework, which we refer to as Adaptive Robot Control using Optimization (ARC-OPT). The framework has been used to generate the results presented in Chapter 4 and 5. ARC-OPT consists of three major parts: The core WBC library, the Rock interface, and the learning module.

The original contributions in this chapter are

- i. A modular software framework, which integrates several WBC approaches on velocity, acceleration, and torque control level in a common interface.
- ii. A learning module that allows us to automatically derive, adapt, and optimize task constraints for WBC.
- iii. An approach for modeling and solving WBC problems on series-parallel hybrid robots.

This chapter is organized as follows: In Section 6.1, we describe the motivation to develop ARC-OPT and its relation to existing WBC software. In Section 6.2 we provide an overview on the software and its integration into existing robot middlewares. Further, we describe the learning module, which can be used to automatically acquire, adapt, and optimize task constraints for Whole-Body Control. In Section 6.3, we introduce a computationally efficient approach for Whole-Body Control of series-parallel hybrid robots, which is integrated in ARC-OPT. Experimental results are provided by evaluating the approach on two different humanoid robots with series-parallel architecture. Finally, we discuss future work and extensions in Section 6.4.

## 6.1 Motivation

Nowadays, several WBC frameworks exist, many of which are available fully or partly as open-source software. Especially the DARPA robotics challenge (DRC) [SBI18] stimulated many developments in this domain and some of the participating groups have published their robot control software in the aftermath of this event. Like most modern WBC approaches, many of the published WBC implementations are based on quadratic programming as described in Section 2.1.3. However, some existing open-source implementations also provide closed-form solutions. Table 6.1 gives an overview on the most popular WBC software frameworks.

The Task Space Inverse Dynamics (TSID) framework, which has first been described by Del Prete et al. [Del+16], implements a control algorithm for legged robots. Like most other WBC frameworks it provides software support for setting up and solving the underlying inverse-dynamics problem, which is formulated as quadratic program. Here, the decision variables are the joint accelerations and the contact wrenches. The framework allows us to specify tasks/constraints in operational and configuration space. As a QP-solver, different variants of Eiquadprog [Buo21] are provided. Although the framework allows us to solve

The ARC-OPT framework is planned to be published as open-source software after dissemination of the scientific results.

Name	Robot Model	Parser	License	Ref.
Task Space Inverse Dynamics (TSID)	Pinocchio	URDF	BSD 2	[Del+16]
Optimization-based framework for Robotic Control Applications (ORCA)	KDL/iDynTree	URDF	CeCILL-C	[Rob21a]
instantaneous Task Specification using Constraints (iTASC)	KDL	URDF	LGPLv2.1 / BSD	[Smi+08]
IHMC Whole-Body Controller	internal	URDF/SDF	Apache 2.0 / GPLv3	[Koo+16]
Drake	internal	URDF/SDF	BSD 3	[TD19]
ControlIt!	RBDL	URDF	LGPL	[Uni21]

Table 6.1: Overview of the most popular open-source software frameworks for WBC

hierarchical QPs, the current solvers do not support task hierarchies. ORCA (Optimization-based framework for Robotic Control Applications) [Rob21a] is a reactive WBC framework that computes the required torques and accelerations for a given set of tasks subject to constraints. As the TSID framework the problem is formulated as single priority QP. Currently the only implemented solver is qpOASES [Fer+14]. The Stack of Tasks (SoT) is a framework that integrates several approaches to control redundant robots, rapidly prototype new applications and verify them in simulation. All controllers are written as optimization problems and include velocity-, acceleration- and torque-level approaches. The SoT also includes different client libraries like Pinocchio [Car+19], a library implementing various rigid body dynamics algorithms. The iTASC framework [Smi+08] is one of the first open-source software frameworks for optimization-based robot control. It implements a hierarchical linear least-squares solver that allows to integrate multiple simultaneously running robot tasks on velocity-level. The iTASC framework is tightly coupled to the Orocos project [BSK03]. The IHMC Whole-Body Controller has been developed for the ATLAS robot [Fen+15] when participating in the DARPA robotics challenge. It provides control algorithms for walking and manipulation based on quadratic programming. The Drake library [TD19] is a collection of libraries for model-based design and control of complex robots. It provides interfaces to several open-source and commercial solvers, including linear least-squares, quadratic programming, and non-linear programming. Finally, ControlIt! [Uni21] is a middleware built around the whole-body operational space control algorithm first introduced by Sentis and Khatib [SK06]. It has been developed at the Human-Centered Robotics Lab of the University of Texas at Austin.

Given the meanwhile long history of Whole-Body Control in robotics, these frameworks provide mature implementations and are rich of features. Still, they lack generality and adaptability, which motivated the development of ARC-OPT. The distinctive features of ARC-OPT compared to the state of the art are listed as follows.

**Usability & Adaptability** Even though the state of the art WBC frameworks presented in this section have produced impressive results on individual robot control problems,

they require human expertise to model the optimization problem and design feedback controllers around it. These handcrafted solutions are often tailored to specific problems and situations. In summary, these frameworks provide pure feedback control algorithms, lacking adaptability and intelligence. Within ARC-OPT, we introduce a Python-based machine learning architecture around a WBC framework, which allows non-experienced users to specify WBC problems by demonstrating the desired robot behavior. Furthermore, using probabilistic adaptation models as described in Chapter 4, the demonstrated whole-body behavior can be adapted to the current context, eliminating the need to repeatedly specify and fine-tune the controllers for every novel situation. Finally, different black-box optimization approaches allow for automatic fine-tuning of task priority functions once the behaviors are deployed on the target robot.

**Generality** Like most other software in the scope of robot motion planning and control, existing WBC frameworks are only able to handle serial or tree-type robot architecture. In contrast, the WBC approaches integrated in ARC-OPT are general in the sense that they can be applied to any robot with serial, tree-type, parallel or hybrid architecture. We elaborate on this feature in detail in Section 6.3, where we introduce a WBC approach for series-parallel hybrid robots.

**Modularity** Most WBC frameworks implement either acceleration/torque-level or velocity-based approaches. While acceleration/torque level approaches are better suited for legged robots with compliant environment interaction, fixed-based systems like mobile manipulators can as well be controlled using velocity-based WBC, which is easier to implement, more stable and less effected by modeling errors. Furthermore, in the industrial domain, most robotic systems do not expose torque control interfaces and velocity-level approaches are the only option. Consequently, both types have their justification. In ARC-OPT we provide a common interface to velocity, acceleration and torque-based WBC approaches for fixed-base and floating-base robots. The framework is developed in a modular way, such that new robot models, solvers and optimization problems can be integrated and easily benchmarked.

## 6.2 Framework Overview

In this section we illustrate the design ideas and key features of the ARC-OPT framework. The framework comprises the core WBC library, an interface to the robotic middleware Rock [RJ21] and the learning module.

### 6.2.1 WBC Library

Figure 6.1 gives an overview of the core WBC library in ARC-OPT and its connection to other components. The library is implemented in C++ with Python bindings for most functionalities. The software design is inspired by the iTaSC framework for constraint-based robot control [Smi+08]. Here, the whole-body controller comprises four separated building blocks, namely controller(s), scene, robot model and solver. In contrast to the iTaSC framework, ARC-OPT defines these modules on library level in order to avoid the dependency to a



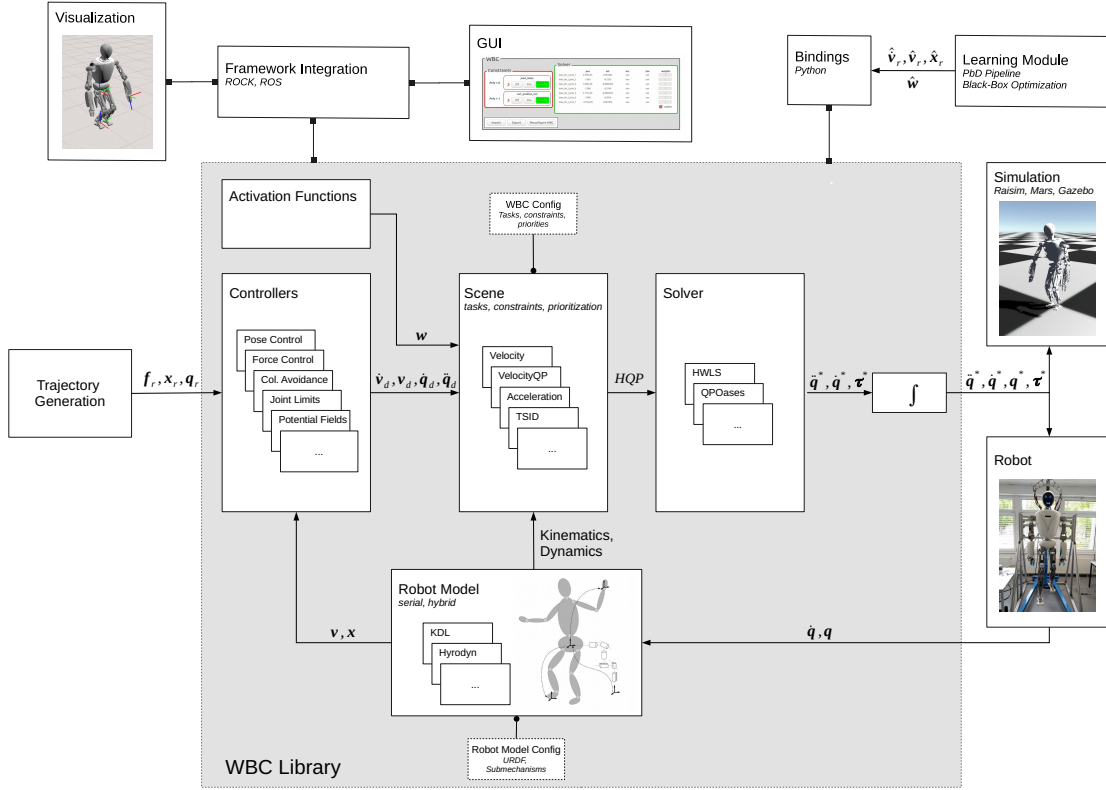


Figure 6.1: Overview of ARC-OPT: Components and their interaction

specific robotic middleware. The functionality of each of these modules is described in the following.

**Controller** A controller implements a task function in operational space, which represents the control objective of a single task, e.g., maintain a certain contact force, follow a trajectory, or avoid an obstacle. The controller can be designed either in task space, in which case it regulates e.g., a desired wrench  $f_r$  or a pose  $x_r$ , or in joint space, where it regulates e.g., a desired joint position  $q_r$ . The control output describes the error of the task function, which is minimized during task execution. Again, the control output is defined either in task space, e.g., a spatial acceleration  $\hat{v}_d$  or a twist  $v_d$ , or in joint space, e.g., a joint acceleration  $\hat{q}_d$  or a joint velocity  $\hat{q}_d$ . Considering the implementation, each controller inherits from a base controller, which assures consistent behavior. This way it is easy to add new controllers. All controllers are agnostic of the robot kinematics and dynamics, as well as the underlying WBC implementation, which brings considerable advantages considering reusability and modularity. Table 6.2 shows an overview of the available controllers.

**Scene** The scene sets up the optimization problem, which is implemented as a variant of the quadratic program in (2.10). The concrete form of the QP can be controlled by a configuration vector comprising one or multiple task configurations. Each task configuration defines the task name, its type (joint or task space), its priority and the task weights, which may influence the relative contribution of the individual task variables within the solution. Furthermore, some robot-specific, semantic information must be provided. For Cartesian

Name	Description	Equation
CartesianPositionController	Cartesian space PD-position/ orientation control with optional velocity/ acceleration feed forward	(2.17), (2.21)
CartesianRadialPotentialFields	Cartesian space radial repulsive potential fields used for dynamic obstacle avoidance	(2.19)
CartesianPlanarPotentialField	Cartesian space planar potential field used for obstacle avoidance	-
CartesianForceController	Cartesian 6 dof force-torque controller	-
JointPositionController	Joint space PD-position/ orientation control with optional velocity/ acceleration feed forward	(2.18), (2.22)
JointLimitAvoidance	Joint position limit avoidance as repulsive control action in joint space	(2.20)
JointTorqueController	Joint space PID force/torque controller	-

Table 6.2: Overview on the implemented controllers in ARC-OPT.

tasks, this information includes the frame of reference, as well as the root and tip frame on the robot defining the (virtual) kinematic chain used to execute the desired task. For joint space tasks, the names of the joints involved must be provided. The tasks are usually formulated within the cost function of a QP, but may also be interpreted as constraints, depending on the implementation of the scene. At runtime, the optimization problem is updated in every cycle with the new reference input from all controllers, the current robot kinematics & dynamics and the task weights, which can be used to (smoothly) activate or deactivate complete tasks or specific task variables. The output of the scene is a (hierarchical) quadratic program (QP), which can be solved using a QP-solver or linear least squares approach. Depending on the implementation of the scene, the QP may allow task hierarchies or implement only a single priority level. Each scene inherits from a common scene interface, which assures consistent behavior. Table 6.3 describes the available scenes including the mathematical formulation of the underlying QPs. Note that the equations are simplified, i.e., they describe QPs with a single task. To deal with multiple tasks, the cost function can be replaced by the weighted sum of the individual tasks as described in (2.15) or implemented as hierarchy as in (2.6).

**Robot Model** The robot model computes the kinematic and dynamic information that the scene requires to set up the optimization problem. This includes different Jacobians and their derivatives, frame transformations, gravity forces and torques, as well as mass-inertia matrices. The robot model is updated in each control cycle with the current joint status of the robot. Currently, two different robot models are implemented:

- **RobotModelKDL** This robot model is based on the Orocos Kinematics and Dynamics Library (KDL) [Pro21], which provides functionality to model and compute kinematics & dynamics of mechanical chains or tree structures. The internal robot representation in KDL is parsed from URDF. While KDL in principle allows to define kinematic loops

Name	Description	Problem Type
VelocityScene	Special case of a QP with linear equality constraints and quadratic cost, i.e., a linear-least squares problem. The decision variables are the joint velocities $\dot{\mathbf{q}}$ .	$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \ \dot{\mathbf{q}}\ _2 \\ \text{s.t.} \quad & \mathbf{J}\dot{\mathbf{q}} = \mathbf{v}_d \end{aligned}$
VelocitySceneQP	Tasks are formulated within the cost functional. Rigid contacts ( $\mathbf{J}_c\dot{\mathbf{q}} = 0$ ) and joint velocity bounds as constraints. The decision variables are the joint velocities $\dot{\mathbf{q}}$ .	$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \ \mathbf{J}\dot{\mathbf{q}} - \mathbf{v}_d\ _2 \\ \text{s.t.} \quad & \mathbf{J}_c\dot{\mathbf{q}} = 0 \\ & \dot{\mathbf{q}}_m \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}_M \end{aligned}$
AccelerationScene	Tasks are formulated within the cost functional. Special case of an unconstrained QP with quadratic cost, i.e., a linear-least squares problem. The decision variables are the joint accelerations $\ddot{\mathbf{q}}$ .	$\min_{\ddot{\mathbf{q}}} \quad \ \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} - \dot{\mathbf{v}}_d\ _2$
AccelerationSceneID	Tasks are formulated within the cost functional. EOM, rigid contacts and joint torque limits as constraints. The decision variables are the joint accelerations $\ddot{\mathbf{q}}$ , joint torques $\boldsymbol{\tau}$ and contact wrenches $\mathbf{f}$ .	$\begin{aligned} \min_{\ddot{\mathbf{q}}, \boldsymbol{\tau}, \mathbf{f}} \quad & \ \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} - \dot{\mathbf{v}}_d\ _2 \\ \text{s.t.} \quad & \mathbf{H}\ddot{\mathbf{q}} + \mathbf{h} = \mathbf{S}\boldsymbol{\tau} + \mathbf{J}_c\mathbf{f} \\ & \mathbf{J}_c\ddot{\mathbf{q}} = -\dot{\mathbf{J}}_c\dot{\mathbf{q}} \\ & \boldsymbol{\tau}_m \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_M \end{aligned}$

Table 6.3: Overview of the available scenes in ARC-OPT

and parallel structures, URDF does not. Thus, this robot model supports only serial or tree-type robots.

- **RobotModelHyRoDyn** This robot model is based on the Hybrid Robot Dynamics (HyRoDyn) framework [Kum+20]. HyRoDyn is a software workbench for computing the kinematics and dynamics of series-parallel hybrid robots. It extends the concepts of URDF by annotating the model with sub-mechanism definitions provided as YAML-files. These sub-mechanism files specify the structure and type of a specific mechanism (e.g., a parallel structure), whose analytical solutions for kinematics and dynamics are implemented in HyRoDyn. Using this robot model, the WBC problem can be set up and solved in the actuation space of series-parallel hybrid robots, a unique feature of ARC-OPT that will be elaborated in Section 6.3. If the given URDF and submechanism file describe a serial or tree-type system, this robot model will produce identical results as the KDL-based model.

**Solver** The solver is a generic component that solves a quadratic program as described in (2.10). Its output is a velocity, acceleration, or torque command in configuration space of the robot. As of now two solvers with strongly different properties are integrated:

- **Hierarchical Least Squares (HLS)** The HLS solver is inspired by a solver implementation integrated in the iTaSC framework [Smi+08]. It solves a sequence of linear least-squares problems by combining the approaches described in (2.6) and (2.7).

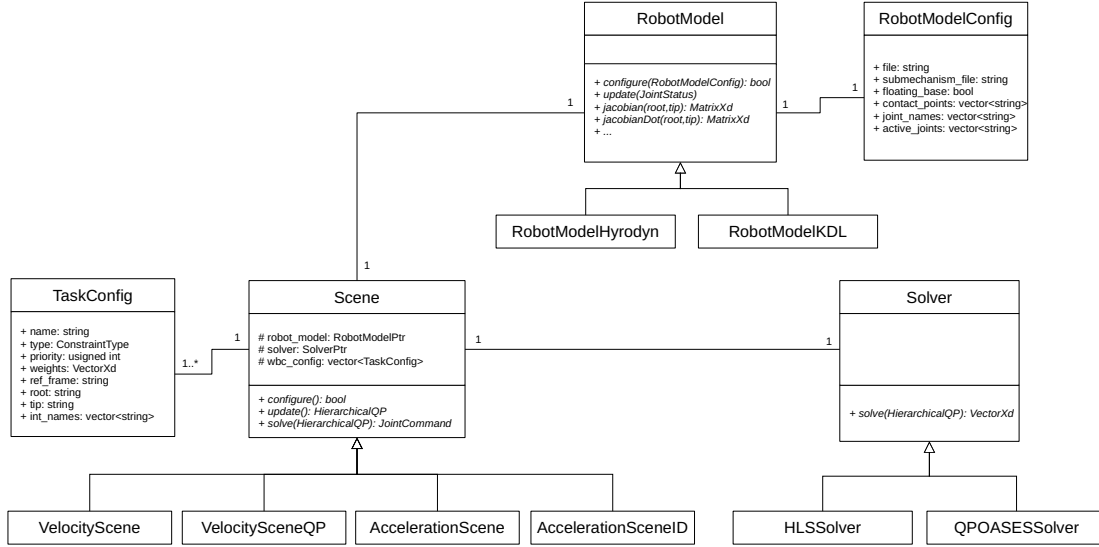


Figure 6.2: Class diagram showing the key components of ARC-OPT

Although (2.6), (2.7) describe a solution on velocity-level, the HLS solver can also be applied to acceleration-based problems. Being a hybrid approach, HLS allows to arrange the tasks in a hierarchy, apply a soft weighting scheme or use a combination of both. In addition, it supports joint weighting in order to prioritize the usage of specific joints of the system. The solution of the hierarchical least-squares problem is computed using a sequence of pseudo inversions with damping term, as described in Section 2.1.1. Note that the least-squares characteristics of this solver does not allow the implementation of hard constraints. This means that even tasks on the highest priority level are only solved accurately if enough dof are available. Otherwise, an approximate solution will be returned, which has the advantage that the solver will never fail to compute a solution due to infeasibility.

- **qpOASES** This solver is a wrapper around qpOASES [Fer+14], which is an implementation of the online active set strategy [FBD08]. It solves quadratic programs as in described (2.10). In contrast to the HLS solver, the qpOASES solver allows hard constraints, like e.g upper and lower bounds of the decision variables. Currently only a single QP can be passed to the solver. Thus, the tasks cannot be arranged in a hierarchy, but only related via a soft weighting scheme. In practice, a two-level hierarchy can be established by implementing the tasks with higher priority as constraints and the tasks with lower priority within the cost functional.

### 6.2.2 Rock Integration

The WBC library described in the earlier section has initially been developed for the use within the Robot Construction Kit (Rock) [RJ21]. Rock is a software framework for the development of robotic systems, which is comparable in scope and usage to other robotic frameworks like ROS [Fou21a] or YARP [MFN06]. It has been developed at the DFKI Robotics Innovation Center (RIC) with the idea of long-term autonomous robots with large scale software systems in mind. In Rock, software libraries are encapsulated in so-called oroGen components, which

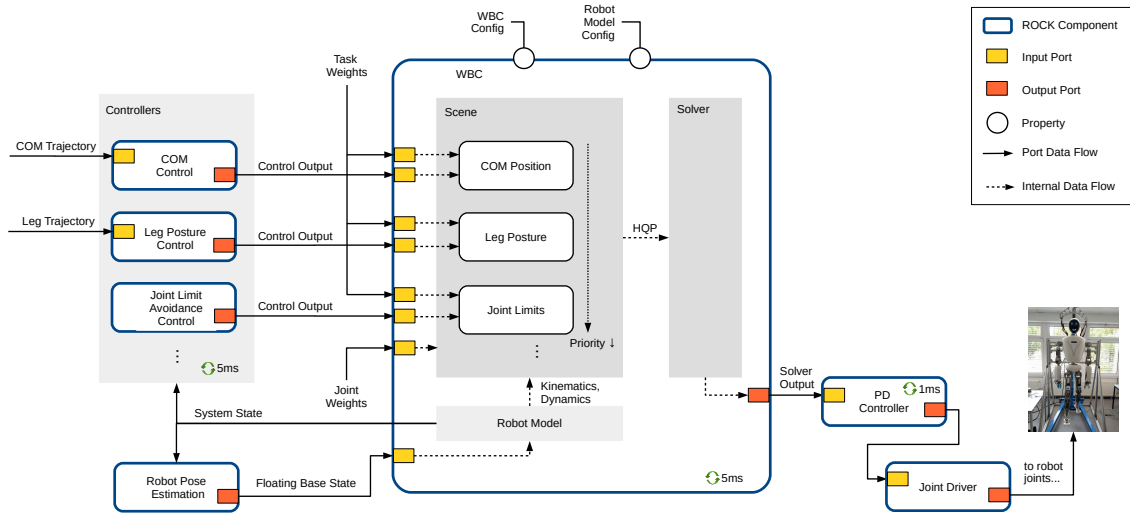


Figure 6.3: Rock integration of the ARC-OPT library exemplified for the RH5 humanoid.

are based on the Orocos Real-Time Toolkit (RTT) [Soe+21]. These components expose the main functionality of the library to other components via communication interfaces like ports (data flow), operations (request-reply) and properties (configuration). Depending on the complexity of the robot and the task, the component network required to control such a system may comprise more than a hundred components.

Figure 6.3 illustrates the Rock integration of the WBC library on the RH5 humanoid robot. The figure only shows the most important components. Each controller is encapsulated in a separate Rock component and thus separated from the main WBC component. This way, arbitrary controllers can be connected to WBC and the existing controller implementations can be reused for other applications. The CoM and the leg posture controller are an implementation of (2.17), while the joint limit avoidance controller is an implementation of (2.20). The input trajectories for the controllers may come either from a motion planner, a trajectory optimization module or they can be learned from demonstration. As mentioned before, the controllers are agnostic to the kinematics and dynamics of the robot and only retrieve the current system state. The control output of each controller is continuously passed to the WBC component. Additionally, the WBC receives the robot state with respect to a global world frame from the robot state estimation module. This module uses an extended Kalman filter to fuse the information from robot forward kinematics, contact force sensors and IMU. It performs a robust estimation of the robot’s base pose, twist, and acceleration with respect to the world frame. As for most of the components in the ARC-OPT framework, the execution of the WBC component is triggered periodically, in this case with a fixed period of 5 ms. In each cycle, it performs the following actions in sequence:

- i. Update the robot model with the current joint state of the robot (position, velocity, and acceleration), including the floating base.
- ii. Set up the optimization problem. Compute Hessian, constraint matrix and bounds for the quadratic program to be solved.
- iii. Solve the QP and write the solver output to the output port.

The WBC passes the solver output (joint position, velocity, acceleration, and torque) to a stabilizing PD-controller. We found that the system is more stable when using an additional



Figure 6.4: WBC configuration example for the RH5 robot.

joint-level PD-controller with feed forward torque, instead of just sending the motor torques to the robot joints and closing the control loop via the WBC. The reason is that, depending on the solver and the size of the problem, the solution of the WBC problem may take 5 – 10 ms, which is too slow for accurate trajectory tracking control (see Section 6.3.4 for results on computation time). The stabilizing PD-controller runs with a cycle time of 1 ms, while the task space controllers run with 5 ms. Depending on the quality of the dynamic robot model, the feed forward torque will compensate for most of the robot dynamics and the PD-gains can be set to low values, which makes the system compliant.

**Configuration Options** The WBC approaches implemented in ARC-OPT can be applied to different robots and a variety of control problems in a flexible manner without having to implement a single line of code. In Rock, configuration properties, which can be loaded from a YAML file, are used to define the runtime behavior of a component. In case of the WBC component, the most important configuration options are the robot model configuration and the task configuration. Figure 6.4 shows an example of a configuration file for the RH5 humanoid. The robot model configuration includes the URDF and HyRoDyn model files and a floating base configuration. In addition, the possible contact points can be defined. The task configuration defines the tasks along with their priorities. Tasks can be defined in joint space (`type: jnt`) or in Cartesian space (`type: cart`). For Cartesian space tasks, WBC requires the definition of three coordinate frames. The `root` and the `tip` frame define the (virtual) kinematic chain, which is used to execute the task, while the reference frame (`ref_frame`) defines the frame in which the control action is performed (see Figure 4.2 for explanation). These frames can be arbitrary links within the entire robot model or external

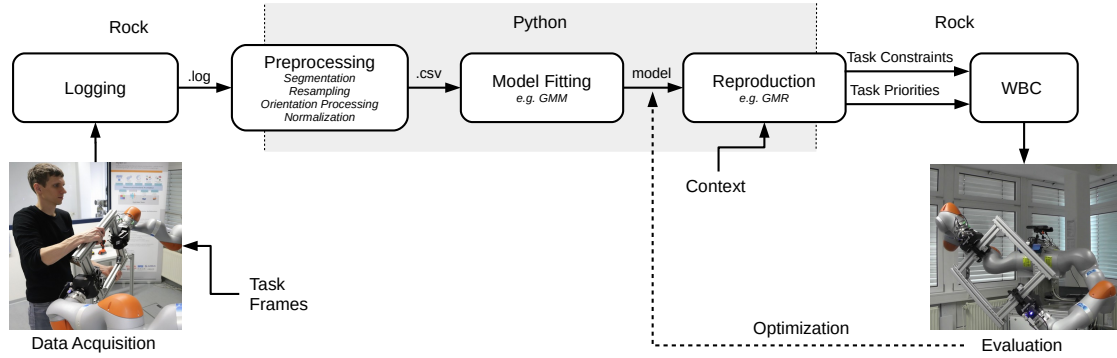


Figure 6.5: PbD pipeline and black-box optimization in ARC-OPT

frames, e.g., an object. For joint space tasks the WBC component requires the names of the joints that contribute to the task. In addition, each task configuration allows the definition of an activation value, i.e., whether the task is initially active or not, and the initial task weights.

By passing different configuration options to the components, the overall behavior can be adapted, even at runtime. A large variety of different control schemes and robot behaviors can be represented through the configuration and composition of different controllers/tasks. For example, it is possible to switch from joint-level control to task space control or from stiff position control to compliant interaction control.

### 6.2.3 Learning Module and PbD Pipeline

To pursue the goal of automatic derivation and contextual adaptation of task constraints for WBC we develop a Python-based architecture for data acquisition, data processing, model fitting, movement reproduction and performance evaluation of whole-body controllers. The architecture connects to the Rock interfaces of ARC-OPT via a Rock-Python bridge, which is a wrapper around the Python implementation of CORBA omniORB. The main functionality of this architecture is to provide a programming by demonstration framework for whole-body behaviors, which allows to specify the optimization problem in WBC without explicit programming. Another aspect of the architecture is to provide methods for black-box optimization of task priorities in WBC. Figure 6.5 shows an overview of the learning pipeline. The workflow is explained in the following.

- **Data Acquisition** For data acquisition we mostly apply kinesthetic teaching, although the input data may be retrieved by any method of choice. In ARC-OPT, constraints are described by the means of task frames, i.e., a constraint describes the relative motion (pose, twist, spatial acceleration) of two coordinate frames, or the wrench applied in a contact coordinate frame. During data acquisition, we record the relative motion between user-defined task frames, as well as the interaction wrenches perceived in these frames. To allow kinesthetic teaching, the robot is required to provide a gravity-mode, which allows guiding the system by hand. Typically, we perform a small number of demonstrations for a certain task and for each context, before we process the acquired data. The context information for each demonstration currently must be provided by



hand. Automatizing recognition and interpretation of the current context, e.g., from the sensor information of the robot, is a non-trivial task and out of the scope of this thesis.

- **Logging** The motion and wrench data is recorded using the Rock logging interface, which stores the data in binary .log-files and provides a software API to inspect, replay and extract the data. The data is converted to .csv files.
- **Preprocessing** After the logging data has been extracted to .csv files, several post-processing steps are applied. At first, we manually segment the demonstrations by identifying feasible start and end points within a plot of the robot's average joint velocity. In future, this manual procedure can be replaced by an automatic segmentation method [GK16]. Afterwards, the data streams are resampled to reduce the amount of data and all streams are temporally aligned. Next, the pose data is processed to provide a singularity-free, continuous, and uniquely defined representation of the orientation between two task frames (see Section 4.4.2 for details). Finally, the entire data set is normalized to have zero mean and unit variance.
- **Model Fitting** After preprocessing, we fit a probabilistic model to the entire data set including context and constraint data. The architecture integrates several approaches for modeling task constraints (GMM, ProMP, ...) and different regression methods for reproducing them in a given context (GMR, GPR, MLP, ...).
- **Reproduction** The task constraints are reproduced in a given context, which currently must be selected by the user. After reproducing the task constraints and their respective priorities, they are sent to the WBC framework via the Rock-Python bridge.
- **Optimization** For optimizing task priority functions, the architecture provides different black-box optimization methods (e.g., CMA-ES or other evolutionary approaches) that use the evaluation feedback from the robot. We implement various loss functions that can be easily compared to each other for a given WBC problem.

The learning module described in this section has been applied to produce the results illustrated in Chapter 4 (PbD pipeline) and Chapter 5 (black-box optimization).

## 6.3 Application: WBC for Series-Parallel Hybrid Robots

In this section, we illustrate the application of the ARC-OPT framework to series-parallel hybrid robots.

### 6.3.1 Motivation

Parallel mechanisms<sup>1</sup> are becoming increasingly popular as subsystems in a variety of robots. In comparison to purely serial or tree-type robots<sup>2</sup>, parallel systems may offer a higher stiffness, payload, and accuracy. On the negative side, such mechanisms have a smaller workspace and are in general more complex to model and control than robots with serial or tree-type architecture. When combining tree-type and parallel robot architectures we obtain series-parallel hybrid robots. They combine the positive aspects of the two types, that is they may provide better dynamic characteristics and a larger workspace compared to serial or

The results presented in this section have been accepted for publication [MSF22]

<sup>1</sup> A parallel mechanism is a system where two or more chains support a single end-effector.

<sup>2</sup> A tree-type robot consists of multiple serial chains connected to a common root link.

tree-type robots. In humanoid robotics series-parallel mechanisms are especially useful in the lower extremities as they may provide a superior payload-to-weight ratio, stiff structure, and optimal distribution of mass-inertia properties. Examples are the ankle mechanisms in NASA's Valkyrie [Rad+15] or in the RH5 humanoid robot of DFKI [Ess+21].

Like most other existing software for robot motion planning and control, the WBC frameworks mentioned in Section 6.1 consider only serial or tree-type robots when modeling the WBC problem. In contrast, parallel or series-parallel hybrid robots, which are designed in a way that they contain parallel loops inside the robot model are mostly not supported. This has two main reasons: On the one hand, robots with serial or tree-type architecture are much easier to model and control than series-parallel hybrid systems. On the other hand, most Whole-Body Control framework applies the Unified Robot Description Format (URDF) to model the robot kinematics and dynamics. URDF is a well-established tool in the robotics community, which, however, forbids the formulation of closed loops inside the model. The common workaround is to abstract parallel mechanisms as one or multiple joints, arranged in series. The kinematics and dynamics of the abstracted parallel mechanisms is usually handled separately in a special function (e.g., [Fou21b]). Such a procedure has several disadvantages:

- ▶ Box constraints<sup>3</sup>, which describe the physical limits of the actuators contained in a parallel mechanism, cannot properly be included in the optimization problem. This way, the admissible workspace and the dynamic properties of the robot may be over- or underestimated.
- ▶ The solution of the optimization problem is usually less accurate since it does not capture the dynamic properties of the parallel subsystems correctly. For example, quantities as the center of mass may be inaccurate when using a serial abstraction of the parallel mechanisms. An extensive study on the effect of neglected dynamics in parallel submechanisms has been presented by Kumar et al. [Kum+19a].
- ▶ The parallel mechanisms may contain singularities that cannot be resolved by the whole-body controller.
- ▶ The resulting control software stack will be more complicated and more difficult to maintain.

Especially the first point is of interest as a proper consideration of the actuator limits in WBC is crucial in terms of an optimal exploitation of the robot workspace. In WBC, the physical limits of the robot actuators, like maximum position, velocity, and torque, are typically expressed as box constraints in the underlying optimization problem. WBC approaches for tree-type robots use serial chains as abstraction of parallel mechanisms, i.e., they hide the parallel mechanism behind independent rotational or prismatic joints. In general, it is not possible to capture the entire configuration space of a robot with closed loops with independent coordinates. Consequently, these approaches cannot consider the actuator limits of joints within parallel systems as box constraints without over- or underestimating the admissible workspace of the robot.

As an example, consider Figure 6.6(a), which illustrates the ankle structure of the RH5 robot. The ankle joint is a 2-dof orientational parallel mechanism of type 2SPRR+1U [Kum+19b]. The actuation principle comprises two linear actuators (ball screw drives), which are attached via two passive rotational joints to the foot link on the lower side and via a passive spherical

<sup>3</sup> Box constraints usually describe an upper and lower bound on the variables of an optimization problem

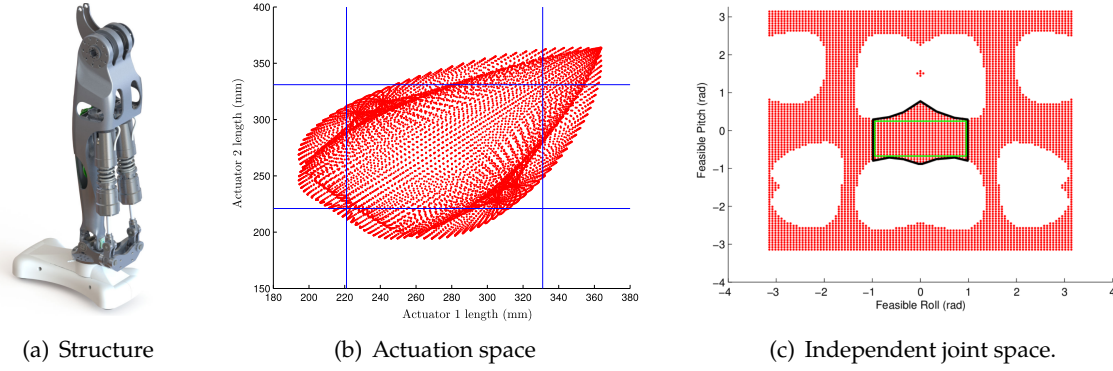


Figure 6.6: RH5 ankle mechanism, implementing box constraints for actuator positions. Image credits [Kum19].

joint to the shank link on the upper side. Moving both actuators in the same direction produces a pure pitch movement of the foot, while moving them in opposite direction will create a roll-movement. Figure 6.6(b) shows the admissible actuator configurations (red dots) of the mechanism. The blue lines represent actuator position limits modeled as box constraints. If we represent this ankle mechanism using independent coordinates (two rotational joints, arranged in series), we achieve the orientation workspace shown in Figure 6.6(c). The black curve denotes the effective workspace after imposing actuator position limits, which is equivalent to mapping the box constraints in actuation space (blue lines in Figure 6.6(b)) into independent joint space. Obviously, it is not possible to model position limits as box constraints in independent joint space without over- or underestimating the effective workspace. Considering safety and reliability aspects we would naturally choose a conservative approximation of the workspace as indicated by the green line. In doing so, we lose part of the admissible workspace and cannot exploit the full capabilities of the robot.

A similar problem can be observed considering box constraints on velocity- or torque-level. Figure 6.7(a) shows the elbow mechanism of the RH5v2 humanoid. The 1-dof elbow structure has a 1-RRPR topology and is a variant of a slider crank mechanism with linear actuator [Kum19]. Figures 6.7(b) and 6.7(c) show that the maximum velocity and torque around the bow of this mechanism are both position dependent. While the maximum elbow velocity can be obtained around the zero configuration where the arm is fully stretched out, the maximum torque can be retrieved when the elbow is inflected by  $-90$  degrees. When modeling velocity/torque limits as box constraints, we naturally select a lower bound as indicated by the dashed line, which ensures that we do not request infeasible velocities or torques from the mechanism. In doing so, we lose part of the admissible velocity or torque range.

To overcome the limitations imposed by tree-type WBC approaches, we introduce a computationally efficient approach for Whole-Body Control of series-parallel hybrid robots. The approach is based on the HyRoDyn library [Kum+20], which is a modular software workbench for computation of kinematics and dynamics of series-parallel hybrid robots. If the kinematic and dynamic quantities needed by WBC are provided by HyRoDyn, we can formulate the optimization problem in the actuation space<sup>4</sup> of a series-parallel hybrid robot. This way, we can consider actuator limits of parallel subsystems as box constraints, in order

<sup>4</sup> The actuation space of a robot is the space of all admissible actuator configurations.

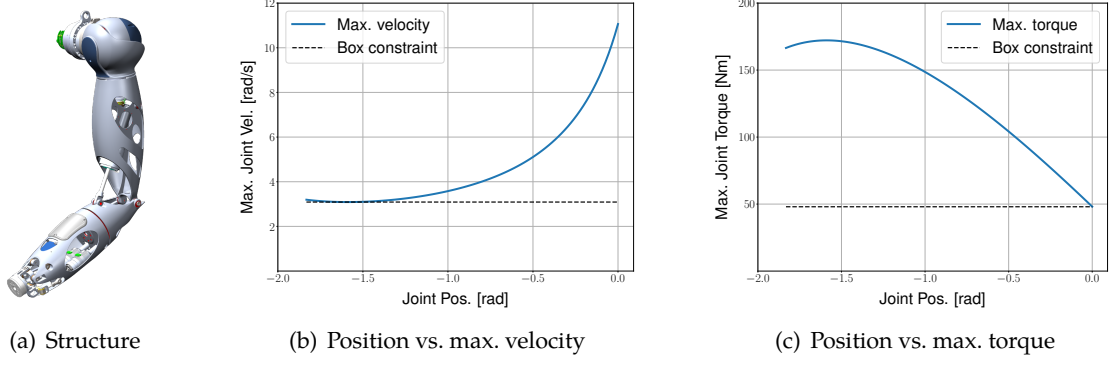


Figure 6.7: RH5 elbow mechanism, box constraints for actuator velocity/torque.

to fully exploit the position, velocity and torque range of all actuators. In contrast, WBC approaches for tree-type robots must apply conservative approximations that underestimate the effective workspace. The approach is integrated in the ARC-OPT framework via the HyRoDyn-based robot model. Thus, the framework is applicable to robots with serial/tree-type, parallel and series-parallel hybrid architecture. We describe the mathematical basics and the actual approach in the following sections.

### 6.3.2 Constrained Kinematics and Dynamics

Robotic systems with closed loops can be described as connectivity graphs, where links and joints are represented as edges and nodes in that graph, respectively. A spanning tree describes a subgraph such that there is one and only one path between any two nodes. Robots with closed loops are subject to loop closure constraints, which are always active and must be resolved in a computationally efficient manner<sup>5</sup>. A robotic system with closed loops may be described by three different sets of coordinates:

- ▶ spanning tree joints  $\mathbf{q} \in \mathbb{R}^{N_n}$ , which describe the entire spanning tree of the robot.
- ▶ independent joints  $\mathbf{y} \in \mathbb{R}^{N_m}$ , also referred to as generalized coordinates, which typically describe a serial abstraction of the parallel subsystems
- ▶ active or actuated joints  $\mathbf{u} \in \mathbb{R}^{N_p}$ , which describe all joints that contain an actuator

Each of these sets constitutes a different space describing the state of the robot, which are denoted as (full) joint space, independent joint space, and actuation space, respectively. Figure 6.8 illustrates the different spaces. The HyRoDyn workbench provides bi-directional mappings between the full joint space, independent joint space, and actuation space, while considering the loop closure constraints of the spanning tree.

### 6.3.3 WBC Approach

In contrast to other WBC approaches, which usually model the optimization problem in independent joint space, we describe the optimization problem in the actuation space of the robot. This allows us to explicitly consider actuator limits of parallel subsystems as box

<sup>5</sup> Note that we distinguish loop closure constraints, which are inherent in the design of a series-parallel hybrid robot, from the task constraints in the WBC problem

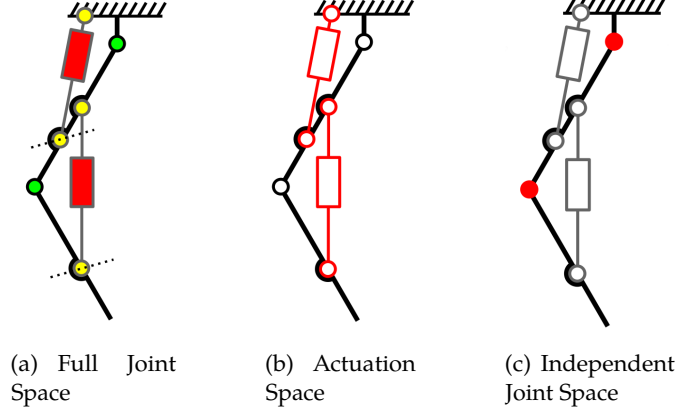


Figure 6.8: Different sub-spaces for modeling series-parallel hybrid robots. Image credits [Kum19]

constraints in the optimization problem. For the sake of simplicity, we consider only a single task in the WBC problem. For velocity-based WBC the proposed approach sets up and solves the following quadratic program:

$$\begin{aligned}
 \min_{\dot{\mathbf{u}}} \quad & \|\mathbf{J}_u \dot{\mathbf{u}} - \mathbf{v}_d\|_2 \\
 \text{s.t.} \quad & \mathbf{J}_{cu}^j \dot{\mathbf{u}} = 0, \quad \forall j \\
 & \dot{\mathbf{u}}_m \leq \dot{\mathbf{u}} \leq \dot{\mathbf{u}}_M
 \end{aligned} \tag{6.1}$$

where  $\dot{\mathbf{u}} \in \mathbb{R}^{N_p}$  are the actuator velocities of the robot,  $\mathbf{v}_d \in \mathbb{R}^6$  is the desired spatial velocity and  $\{\dot{\mathbf{u}}_m, \dot{\mathbf{u}}_M\}$  are the actuator velocity limits. The Jacobian  $\mathbf{J}_u \in \mathbb{R}^{6 \times N_p}$  maps actuator to task space velocities and is related to the Jacobian  $\mathbf{J} \in \mathbb{R}^{6 \times N_n}$  of the spanning tree as follows:

$$\mathbf{J}_u = \mathbf{J} \mathbf{G} \mathbf{G}_u^{-1} \tag{6.2}$$

Here,  $\mathbf{G} \in \mathbb{R}^{N_n \times N_m}$  is the loop closure Jacobian that relates independent to spanning tree coordinates  $\dot{\mathbf{q}} = \mathbf{G} \dot{\mathbf{y}}$ . Furthermore,  $\mathbf{G}_u \in \mathbb{R}^{N_p \times N_m}$  is the actuator Jacobian, which comprises the rows of  $\mathbf{G}$  that correspond to the actuated joints:  $\dot{\mathbf{u}} = \mathbf{G}_u \dot{\mathbf{y}}$ . The derivation of loop constraints shall be skipped here for brevity, but algorithms to compute them can be found in standard textbooks [Fea14; Jai10].

On acceleration level, we derive the following WBC problem in actuation space:

$$\begin{aligned}
 \min_{\ddot{\mathbf{u}}, \tau_u, \mathbf{f}} \quad & \|\mathbf{J}_u \ddot{\mathbf{u}} + \dot{\mathbf{J}}_u \dot{\mathbf{u}} - \dot{\mathbf{v}}_d\|_2 \\
 \text{s.t.} \quad & \mathbf{H}_u \ddot{\mathbf{u}} + \mathbf{h}_u = \tau_u + \sum_j \mathbf{J}_{cu}^j \mathbf{f}_j \\
 & \mathbf{J}_{cu}^j \ddot{\mathbf{u}} = -\dot{\mathbf{J}}_{cu}^j \dot{\mathbf{u}}, \quad \forall j \\
 & \tau_{um} \leq \tau \leq \tau_{uM}
 \end{aligned} \tag{6.3}$$

Here,  $\ddot{\mathbf{u}}, \tau_u$  are the accelerations and forces/torques of the actuators,  $\mathbf{f}_j$  are the contact wrenches and  $\dot{\mathbf{v}}_d$  the desired spatial acceleration. Again, we assume a single task and fully

actuated robot for the sake of simplicity. The tasks are solved respecting the EOM in actuation space, rigid contacts, and actuator force/torque limits  $\{\tau_{um}, \tau_{uM}\}$ . The dynamic quantities in actuation space required to set up this QP are again computed using the loop closure Jacobian. For example, the mass-inertia matrix in actuation space  $\mathbf{H}_u$  can be derived from the mass-inertia matrix of the full spanning tree  $\mathbf{H}$  as follows:

$$\mathbf{H}_u = \mathbf{G}_u^{-T} \mathbf{G}^T \mathbf{H} \mathbf{G} \mathbf{G}_u^{-1} \quad (6.4)$$

Through the ability of describing optimization problems in actuation space of series-parallel hybrid robots, the proposed WBC framework is more general than the existing approaches as it can be applied to any kind of robot without the need to abstract important details. In the following section we demonstrate the application of the approach to different series-parallel hybrid robots and compare its performance with respect to WBC approaches that solve the optimization problem in independent joint space.

Note that (6.1) - (6.4) only apply if the robot is fully actuated, that is  $N_p = N_m$ . To handle robots with a floating base, we have to replace the configuration vector of the actuated joints  $\mathbf{u}$  with  $\tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{u}_b^T & \mathbf{u}^T \end{pmatrix}$ . Here,  $\mathbf{u}_b \in \mathbb{R}^6$  is the configuration of the virtual 6-dof linkage representing the floating base. Furthermore, all kinematic and dynamic quantities used in the QP must be replaced by their counterparts that include the floating base. This includes all Jacobians  $\mathbf{J}_u$  and  $\dot{\mathbf{J}}_u$ , the mass-inertia matrix  $\mathbf{H}_u$  and the bias force/torque term  $\mathbf{h}_u$ .

### 6.3.4 Results

This section presents experimental results to demonstrate the ability of the proposed approach to exploit the entire position, velocity, and torque workspace of series-parallel hybrid robots. Also, we study the computational performance of the WBC architecture here. Experiments are performed using the RH5 and RH5v2 humanoid robots, in simulation as well as on the actual systems. As a benchmark, we compare the results to an analogue WBC approach for tree-type systems.

#### Application of Box Constraints in Actuation Space

As described in Section 6.3.1, integration of actuator constraints (e.g., maximum actuator positions, velocities or torques) for parallel mechanisms is usually not possible in existing WBC frameworks as they use an abstract representation of the mechanical structures. The abstract workspace cannot be captured entirely using box constraints. Thus, a conservative approximation of the reachable workspace must be selected in order to avoid deadlock situations or even mechanical damage to the actuation system. This approximation usually leads to a reduced workspace as depicted in Figures 6.6 and 6.7. In contrast, describing the WBC problem in actuation space allows to exploit the full workspace. This is demonstrated in the following.



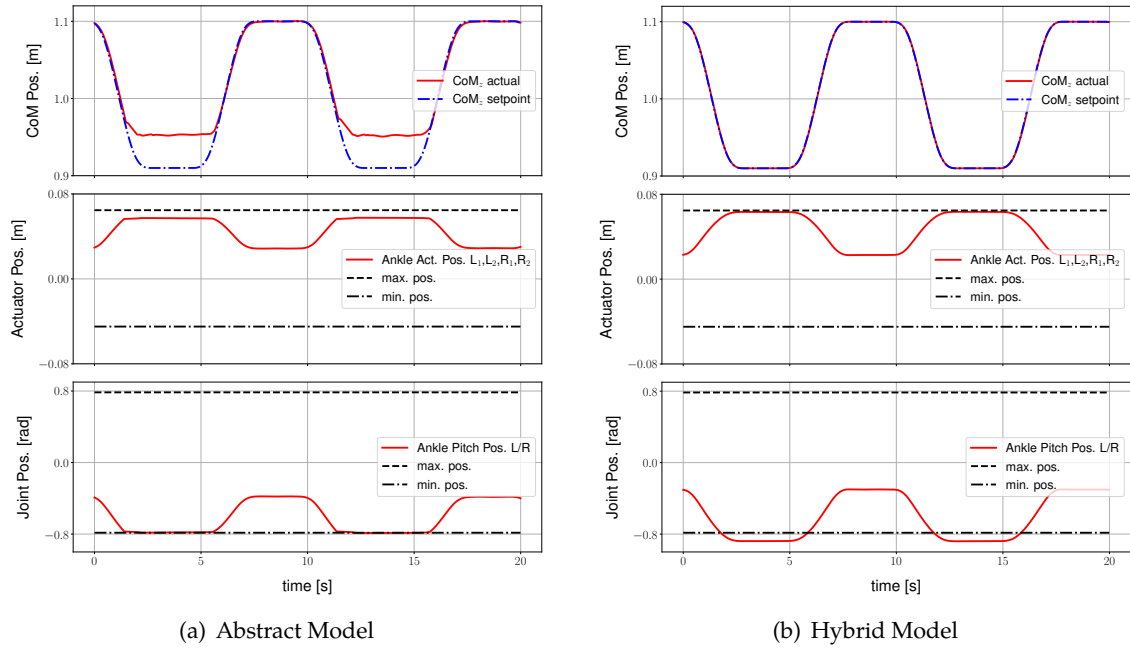


Figure 6.9: Comparison of squatting motion on RH5 using WBC with abstract/tree-type model and series-parallel hybrid model.

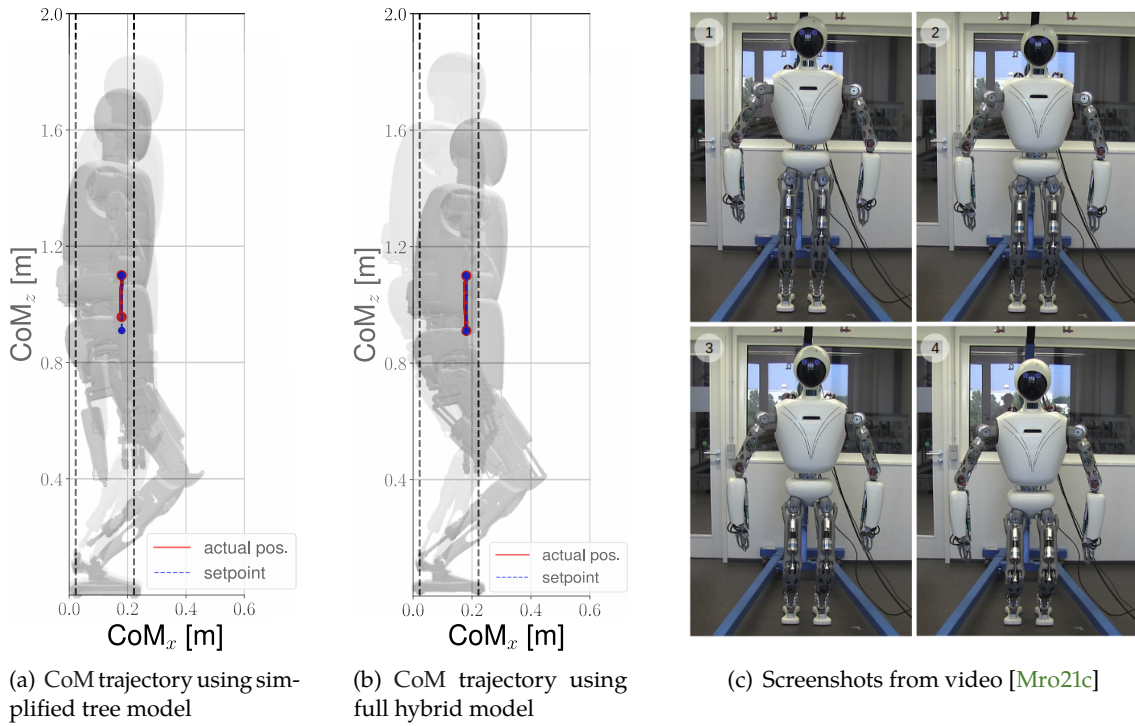


Figure 6.10: Squatting motions on the RH5 humanoid.



**Position Constraints** We evaluate the application of position limits as box constraints in actuation space using the proposed WBC approach for series-parallel hybrid robots. Experiments are performed on the RH5 humanoid (see Figure 1.3) using velocity-based WBC as in (6.1). We implement two tasks, namely CoM control enforced by a Cartesian position controller as in (2.17) and joint limit avoidance using repulsive potential fields as described by (2.20). We program squat movements by translating the CoM  $\pm 0.2m$  in z-direction, which is oriented vertically to the ground. Considering the desired squatting motion, the ankle mechanism of RH5 as depicted in Figure 6.6(a) is particularly important. The position limits of the ankle mechanism in actuation space and in independent joint space are as follows:

	Joint Name	Upper	Lower
Independent	Ankle Pitch L/R [rad]	0.5236	-0.5236
Joint Space	Ankle Roll L/R [rad]	0.7850	-0.7850
Actuation Space	Ankle Act $L_1, L_2, R_1, R_2$ [m]	0.0647	-0.0449

Figure 6.9 compares the squatting motion of our approach with the analogue WBC approach for tree-type robots. The plots illustrate the CoM trajectory (top), the ankle actuators (middle) and the independent ankle pitch joint (bottom). For squatting, the ankle joints follow a pure pitch trajectory and all 4 ankle actuators  $L_1, L_2, R_1, R_2$  move identically. Figure 6.10 shows the motion of the CoM in the xz-plane. Using the hybrid WBC approach, we can obtain deeper squat movements as the entire actuation space is exploited. In contrast, when using the tree-type WBC approach, the admissible workspace of the actuators is reduced as we have to define conservative position limits in independent joint space, which again compromises the CoM trajectory. In the experiments, we can perform squatting movements in a range of  $0.957 - 1.100m$  on the z-axis using tree-type WBC, while achieving a range of  $0.911 - 1.100m$  using the hybrid WBC approach. Using the proposed approach in actuation space we can fully exploit the capabilities of the robot, gaining about  $4.5cm$  or approx. 25% of admissible workspace.

**Velocity Constraints** Next, we evaluate the formulation velocity limits as box constraints in actuation space using the RH5v2 robot (see Figure 1.3) as experimental platform. RH5v2 is a series-parallel hybrid system, including multiple closed-loop kinematic structures. We execute highly dynamic boxing movements, which are generated using offline trajectory optimization based on Differential Dynamic Programming (DDP) [Mas+20]. The motions are executed using the velocity-level WBC approach in (6.1). We focus on the elbow mechanism of the robot, whose maximum velocity is configuration dependent, as shown in Figure 6.7. Using a tree-type WBC approach, we would necessarily select conservative velocity limits as box constraints in independent joint space, which underestimate the velocity range of the mechanism. In contrast, using our WBC approach, we can define velocity limits in actuation space. The velocity limits of the elbow mechanism in independent joint space and actuation space are:

	Joint Name	Upper	Lower
Ind. Joint Space	Elbow L/R [rad/s]	3.09	-3.09
Actuation Space	Elbow Act L/R [m/s]	0.266	-0.266

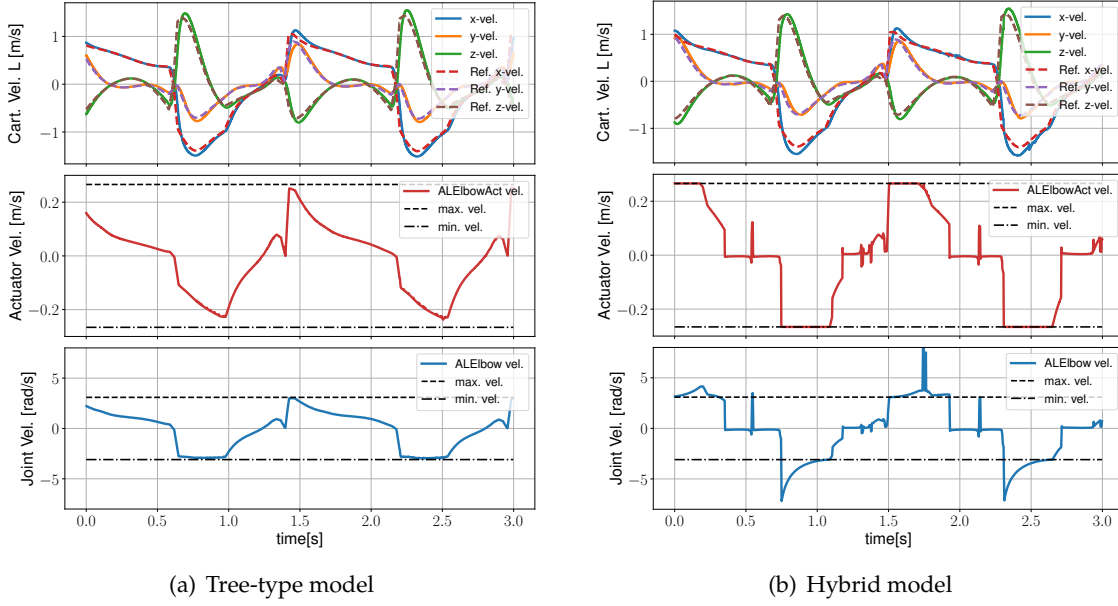


Figure 6.11: Comparison of boxing motions on RH5v2 using WBC with abstract/tree-type model and series-parallel hybrid model.

Figure 6.11 shows a comparison between the resulting boxing motions using simplified tree-type and full hybrid WBC approach. The plots show the linear end-effector velocities of the left arm (top), the linear actuator velocity of the left elbow mechanism (middle) and the radial elbow motion in independent joint space (bottom). The maximum velocity of the linear actuator cannot be exploited when choosing conservative velocity limits in independent joint space, which is the case for the tree-type WBC approach. In contrast, the hybrid approach is able to fully exploit the velocity range and achieves independent joint space velocities greater than 3.09 rad/s, which is the conservative maximum in independent joint space. The linear end-effector velocities are identical in both cases, as the other arm joints compensate for the reduced elbow velocity in the tree-type case. Nevertheless, the maximum independent joint velocity of the elbow mechanism is about twice as high in the hybrid case. Thus, the hybrid approach allows for more dynamic movements than the corresponding tree-type WBC approach. Figure 6.12 shows snapshots from a video of executing boxing motions on the RH5v2 robot.

**Force/Torque Constraints** Although being skipped here for the sake of brevity, we expect similar results on torque-level, i.e., when describing maximum actuator forces/torques of a series-parallel hybrid robot as box constraints in a whole-body controller. In particular applications from humanoid robotics like walking or jumping are highly dynamic and require an optimal exploitation of the feasible actuator forces and torques.

### Computational Performance

In this section, we evaluate the computational performance of the proposed WBC framework. We measure the computation time for a complete update cycle, which comprises the computation times for updating the QP and solving it using qpOASES. All experiments

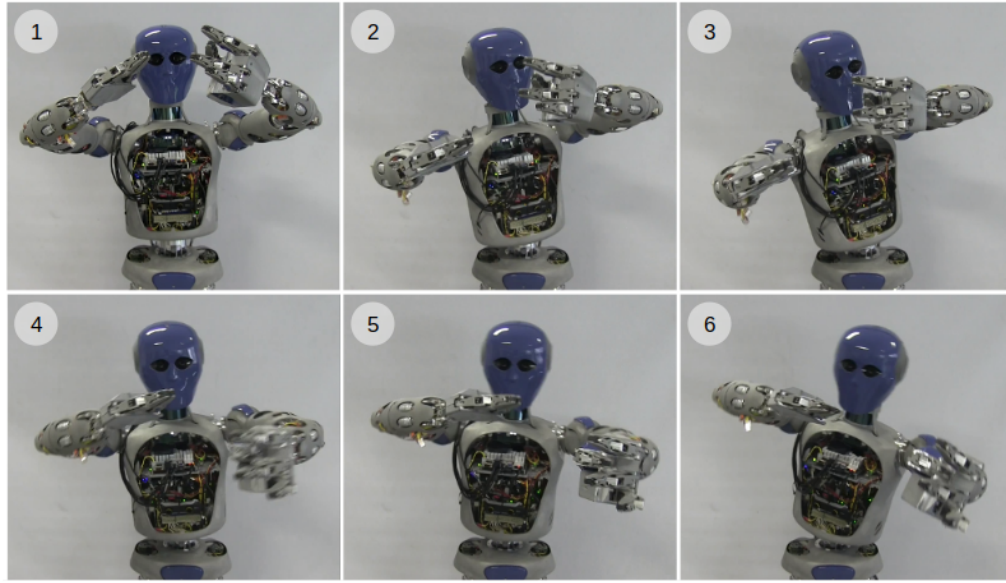






Figure 6.12: Executing boxing motions on RH5v2, screenshots from video [Mro21c].

#	Robot Type	Model Type	WBC Type	Model Size			QP Size		
				$N_n$	$N_m$	$N_p$	$N_q$	$N_c$	
1	RH5v2*	serial	(6.1)	20	20	20	20	0	
2		hybrid		61	20	20	20	0	
3		serial	(6.3)	20	20	20	40	20	
4		hybrid		61	20	20	40	20	
5	RH5 One Leg*	serial	(6.1)	6	6	6	6	0	
6		hybrid		18	6	6	6	0	
7		serial	(6.3)	6	6	6	12	6	
8		hybrid		18	6	6	12	6	
9	RH5 Both Legs**	serial	(6.1)	18	18	12	18	12	
10		hybrid		42	18	12	18	12	
11		serial	(6.3)	18	18	12	42	30	
12		hybrid		42	18	12	42	30	
13	RH5**	serial	(6.1)	38	38	32	38	12	
14		hybrid		83	38	32	38	12	
15		serial	(6.3)	38	38	32	82	50	
16		hybrid		83	38	32	82	50	

\*\*Fixed Base, \*Floating Base

Table 6.4: Complexity of different robots models.

are performed in a simulation environment provided by the RaiSim multi-body physics engine [HLH18], running on a standard Laptop with an Intel Core i7-8565U CPU (8 x 1.8GHz). To examine the effect of an increasing model complexity on the computational effort, we use different robot models as illustrated in Table 6.4. Here  $N_n, N_m, N_p$  are the number joints contained in the spanning tree, the independent joints, and actuators. For the serial models,  $N_n = N_m$ , that is the spanning tree and the independent joint space match. Furthermore, the employed models without floating base are all fully actuated, that is  $N_m = N_p$ . For the models that include a 6 dof floating base, the independent joint space is larger than the actuation space. The QP problem size is described by  $N_q$  and  $N_c$ , which are the number of decision variables (joint velocities for the velocity-based WBC, joint accelerations, torques and contact wrenches for the acceleration-based WBC) and the number of constraints, respectively. Both  $N_q$  and  $N_c$  depend on the complexity of the robot model, the WBC implementation and the problem that is to be solved. Here, we use a WBC implementation on velocity-level as in (6.1) and a WBC implementation on acceleration/torque level as in (6.3), which provide different QP sizes. As an example, consider #16 from Table 6.4, which shows the model complexity of the series-parallel hybrid (full) RH5 model for the dynamic acceleration/torque-based WBC approach. In this model, the robot has 32 actuators in total and 38 independent joints. The latter include a virtual 6-dof linkage for the floating base, which has a spatial acceleration  $\dot{\mathbf{v}}_b$ . Furthermore, we have two rigid contacts of the feet with the ground floor, which are expressed as 6-dimensional no-slip constraints. In total, we get the following decision variables:

$$\begin{aligned} \tilde{\mathbf{u}} = ( \dot{\mathbf{v}}_b \quad \ddot{\mathbf{u}} )^T &\in \mathbb{R}^{38} \\ \boldsymbol{\tau}_u &\in \mathbb{R}^{32} \\ \mathbf{f}_j &\in \mathbb{R}^6, j = \{1, 2\} \end{aligned} \quad (6.5)$$

and the constraints:

$$\begin{aligned} \mathbf{H}_u \ddot{\mathbf{u}} + \mathbf{h}_u &= \boldsymbol{\tau}_u + \sum_j \mathbf{J}_{cu}^j \mathbf{f}_j \in \mathbb{R}^{38} \\ \mathbf{J}_{cu}^j \ddot{\mathbf{u}} &= -\dot{\mathbf{J}}_{cu}^j \dot{\mathbf{u}} \in \mathbb{R}^6, j = \{1, 2\} \end{aligned} \quad (6.6)$$

This gives in total  $38 + 32 + 2 \cdot 6 = 82$  decision variables and  $38 + 12 = 50$  task constraints. In addition to these constraints, every QP is subject to  $N_p$  actuator limits modeled as upper and lower bounds on the decision variables. However, these qpOASES handled these decision bounds much faster than actual constraints, so we do not mention them in Table 6.4. Considering the robot task used for experimental evaluation, we define similar tasks for all robot models. For the RH5v2 robot, we specify a 6 dof Cartesian task for each gripper. For the RH5 single leg robot, we specify a 6-dimensional task for the ankle link. For the floating base RH5 legs and the full RH5 robot model, we specify a 6-dimensional Center of Mass tracking task, where the foot links are subject to rigid contact constraints.

We compare the computation time for solving the same WBC problem on a series-parallel hybrid robot and on the corresponding simplified tree-type model. The results are shown in Figure 6.13. We found that the computation time using the hybrid models is around 1.2 – 2.5 larger on average. Since we describe the optimization problem in actuation space, the QP size is identical for both cases (considering the same robot model and WBC type) as can be seen in Table 6.4. However, the series-parallel hybrid robot models have a larger number of joints in total (spanning tree). Also, the consideration of the loop closure constraints requires additional computational effort. For simple models, this effect can be neglected.

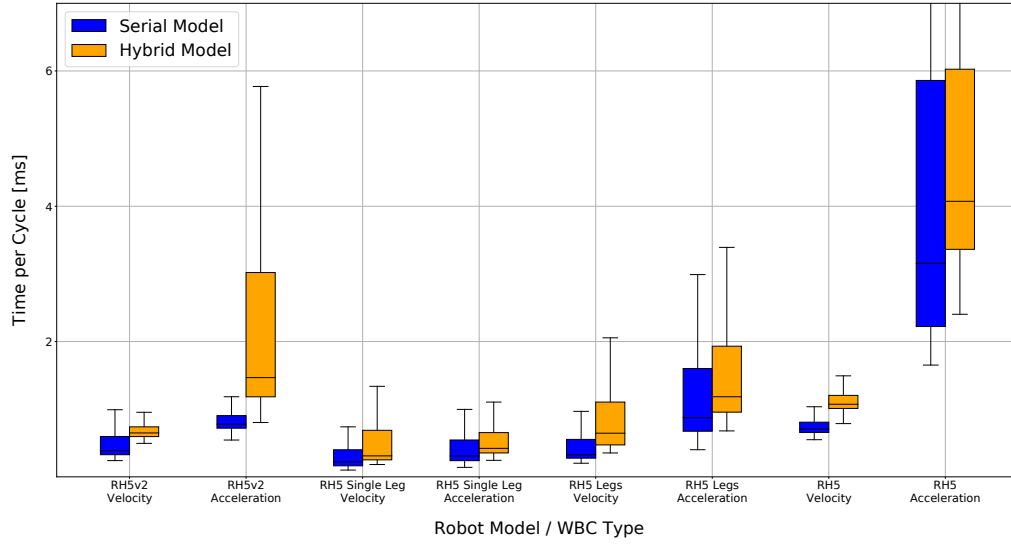


Figure 6.13: Comparison of serial vs. hybrid model regarding the computation time for a complete cycle (model update, scene update, QP solving) for different robot models and scenes.

For increasingly complex models, the difference in computation time between serial and hybrid model increases as well. The additional computational effort is justified given to the advantages we get regarding model fidelity. Anyways, the approach is still able to control the RH5 system, which has the most complex robot model, with a cycle time well below 10 ms.

Still, a further reduction of the computation time can be achieved by various measures. First, one could avoid the use of a numerical QP solver. Switching to an analytical least-squares approach not only reduces the computation time, but it also makes it easier to assume an upper bound for the required computational effort. In contrast, a numerical solver may take arbitrary long to find a solution in case of infeasibility. However, this way one loses the capability of describing inequality constraints. Further, least-squares approaches only allow soft constraints as they return the best-possible nearby solution, while QP solvers can also deal with hard constraints. Another possibility to decrease the computational effort is to reduce the QP problem size, for example by expressing  $\tau_u$  as a function of  $\ddot{\mathbf{u}}$ . This way one could eliminate  $\tau_u$  as a decision variable. The inverse dynamics solution, if required, could be computed in a subsequent step using a specialized solver. Finally, one could reduce the model complexity in an intelligent way. For example, one could use the full model, where heavy masses are moved inside the parallel submechanisms and use an abstract model for the lightweight constructions. In [Kum+19a], the authors argue that simplified series-parallel hybrid models can be used in some cases, which speeds up computation by more than 50%, while providing similarly accurate results as the full model.



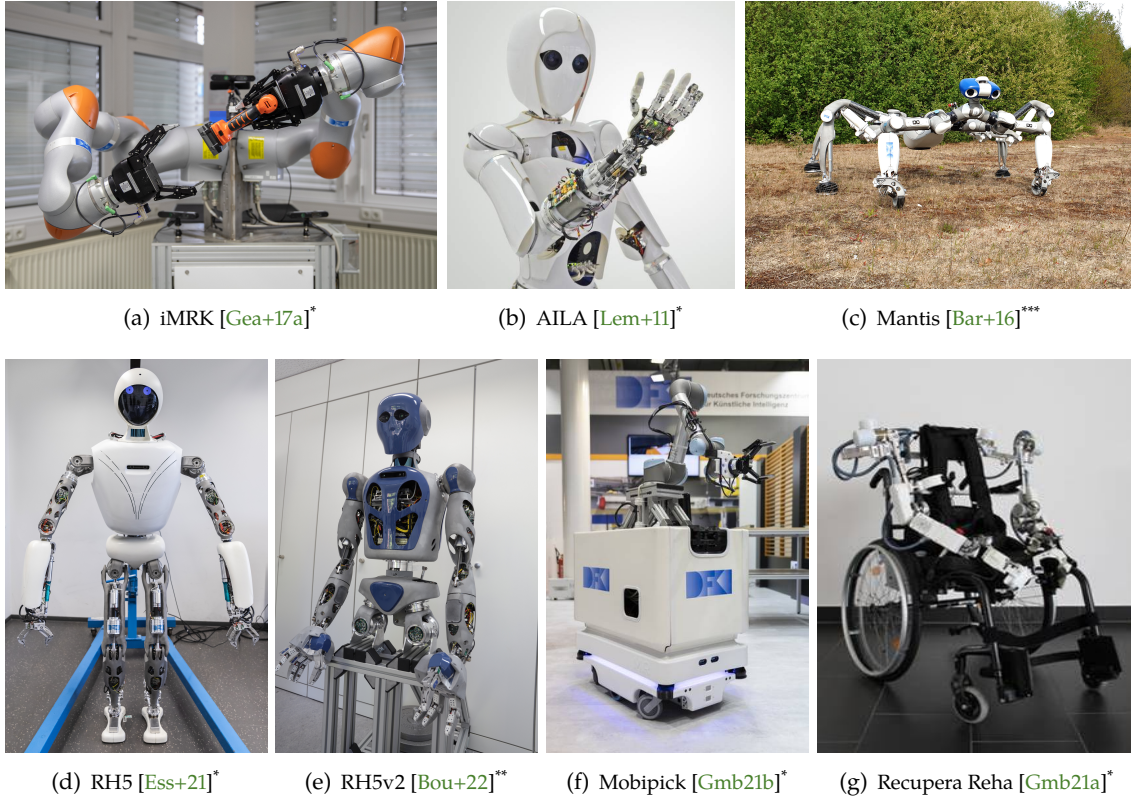


Figure 6.14: Robotic systems at DFKI RIC that the ARC-OPT framework has been applied to.  
Image Credits: <sup>\*</sup> Annemarie Popp, <sup>\*\*</sup> Thomas Frank, <sup>\*\*\*</sup> Alexander Dettmann (all DFKI)

## 6.4 Discussion

In this chapter we introduced ARC-OPT, which is a modular software framework that provides various WBC implementations on velocity-, acceleration- and torque-level. The framework integrates a novel WBC approach for series-parallel hybrid robots, which makes ARC-OPT more generally applicable than existing frameworks. The WBC core of ARC-OPT is integrated into a Python-based architecture for learning task constraints and their respective priorities from user demonstrations. This allows us to automatically derive task constraints from data, adapt them to different application contexts and optimize the task priorities once the solution is deployed on the target robot.

Figure 6.14 shows different robotic systems at DFKI RIC, which are currently or have been controlled by ARC-OPT or by one of its predecessor frameworks. These robots include industrial manipulators, mobile systems, multi-legged robots, humanoids and even an upper-body exoskeleton used for teleoperation. The heterogeneity of these systems shows the general applicability of ARC-OPT. To increase visibility outside DFKI and receive valuable community feedback, the framework will be published as open-source software after the scientific dissemination of the results.

The core of ARC-OPT, which is the whole-body controller with different configuration options considering solver, robot model and control output type is mature and well-tested. However, within the Python architecture for learning and optimizing task constraints, we envisage

several future steps to improve workflow and generality of the approach. First, there are still various manual steps included in the workflow, which shall be replaced by automatized procedures soon. For example, the demonstrated movements are currently segmented by the user with the help of a simple graphical tool, which is time-consuming. To improve this situation, one could either apply an automatized segmentation process or attempt to fully avoid the need to segment the demonstrated tasks. The latter can be achieved by means of suitable user interfaces for starting data acquisition, e.g., a button attached to the robot, or any other means to detect start and stop of the demonstrated movement. Furthermore, the user currently must define the given context by providing a real-valued context vector. An automatized procedure for context classification could be based on unsupervised learning approaches that assign categories to the acquired data after all demonstrations have been collected. At execution time, the current sensory information can be used to select the category and assign the corresponding task constraints. While context classification is a non-trivial problem in general in robotics, for a finite number of discrete contexts it is known to work well. Examples can be found in contact classification in human-robot collaboration [PKM17; HDA17].



This chapter recapitulates the main contributions of this thesis and draws conclusions on the results. Furthermore, an outlook on future work is given.

## 7.1 Summary and Discussion

This thesis is motivated by the fact that Whole-Body Control, though being a powerful tool to specify and control complex robot tasks, requires a human expert to analyze the task, specify task constraints and assign suitable task priorities. Apart from being time-consuming and error-prone, this procedure results in solutions that lack generality. Once the situation or the task changes, the specification must be adapted, which increases the general effort for modeling and reduces the autonomy of the robot as the human operator has to intervene. If we demand robotic systems that operate autonomously over a long period, we require adaptive control approaches that consider the current context and reason about suitable, situation-specific controller parameters.

In order to serve these needs, we introduce a programming by demonstration approach for whole-body controllers in this thesis. The approach, which is based on GMM-GMR allows to automatically derive parts of the optimization problem in WBC from data acquired in user demonstrations. By demonstrating a robot task in varying situations, we can generalize the acquired knowledge and adapt the whole-body controller with respect to the current situation, which is referred to as context. These generalization capabilities may help to develop robots that perform better in dynamic environments, act more autonomously and require less human intervention and expertise. We demonstrate the capabilities of the approach regarding different manipulation tasks on the iMRK system, an industrial dual-arm robot, and on the RH5 humanoid. It is shown that the approach is able to automatically adapt task constraints and priorities with respect to a limited number of contexts, where the context description may be discrete, as in "rotate right", "rotate left", or continuous, e.g., the size of a manipulated object. Furthermore, it is demonstrated that the approach performs better than state-of-the-art methods in terms of reproduction error for previously unknown contexts. Finally, we develop an alternative PbD approach based on ProMPs and compare its performance to the GMM-GMR-based method.

Once being deployed on the target robot, there is no guarantee that the learned behavior is optimal as it depends on the quality of user demonstrations. Therefore, this thesis also investigates black-box optimization approaches for task priorities and other WBC parameters in order to increase the quality of the obtained solutions. The developed methods are evaluated and compared using the iMRK robot. It is shown that we can use black-box optimization to improve robot behavior in terms of accuracy and smoothness.

The contributions are aggregated in a modular and adaptive Whole-Body Control framework named Adaptive Robot Control using Optimization (ARC-OPT). The framework aids at programming complex robots like humanoids, multi-legged systems, or mobile manipulators

by combining machine learning methods and WBC. We demonstrate one of the core features of this framework, namely its general applicability, on two different humanoid robots with series-parallel hybrid structure.

The combination of Whole-Body Control and programming by demonstration has immense potential. While PbD offers an end-user interface to intuitively program new robot tasks, WBC provides a powerful tool for experts to specify complex behaviors for redundant robots. The integration of the two approaches provides improvements in terms of usability, general applicability, and autonomy of robots in dynamic environments. Also, it is possible to integrate expert knowledge by programming some tasks in a manual fashion and learning other tasks that are too hard to program.

Naturally, creating autonomous robot behavior requires more than intelligent control techniques as described in this thesis. In our work we assume that the context in which the robot is currently operating is known. Estimating the current operational context with high certainty is a crucial ingredient in robotic decision making, but hard to implement in complex environments. Furthermore, the experimental evaluation in this thesis mostly focuses on individual robot tasks. To generate complex robot behavior, we require approaches for planning, executing, and monitoring sequences of tasks on a symbolic level. Although we do not touch the area of task planning, the approaches presented in this thesis facilitate high-level planning as they are able to select the correct controllers for a certain situation. Thus, the approaches have the potential to bridge the gap between high-level symbolic planning and numerical control.

## 7.2 Future Work

This thesis addresses certain limitations of WBC approaches in order to make them more user-friendly, general and performant in dynamic environments. The adaptive WBC approaches presented in this thesis provide a basis for future lines of research, which shall be summarized in the following.

- **Contact Interaction Tasks** Modeling time-varying contact situations is one of the key problems to be solved when controlling a humanoid robot. Complex interaction tasks, as also required in dexterous object manipulation, are difficult to model. To compensate for inevitable modeling errors and further reduce the effort for the human expert, the PbD approach presented in Chapter 4 could be extended to the estimation of contact constraints in future. Both, location of an environment contact and the time of contact making could be derived from user demonstrations and generalized to varying situations using machine learning approaches. In future, we plan to increase our efforts in this regard and integrate contact constraints derived from user demonstrations with other tasks in the WBC problem.
- **Context Identification** In order to create autonomous robot behavior, it is required that the robot perceives the environment through sensors and identifies the context in which it is currently operating from the acquired sensor data. In relation to the approaches presented in this thesis, a relevant line of research would thus be to automatize the process of context identification, i.e., assign context labels given the data acquired from user demonstrations, distinguish known and unknown contexts, identify the current

context during operation of the robot and select the appropriate whole-body controller based on this classification. Such an automated procedure should replace the manual context definition and identification used in the approach presented in this thesis on the long run.

- **Link to Symbolic Task Planning** The focus of this thesis is to create more intelligent WBC approaches. We create context-adaptive feedback controllers for redundant robots. The area of high-level task planning and execution is not touched here, although the implemented approaches may provide a helpful link to symbolic robot control. Thus, a natural extension would be to create task sequences using symbolic planning and use ARC-OPT as execution layer, which is able to select optimal WBC parameters for a certain situation.
- **Model Selection** The experimental evaluation presented in Chapter 4 focuses on a finite number of tasks and contexts obtained in user demonstrations. When we deploy the approaches presented in this thesis in real-world applications, the number of tasks, contexts and derived adaptation models may grow quickly. Thus, an intelligent way to manage the generated models is required. Questions like, "Which model to select for a given task?", "How to evaluate model performance?" and "When should I re-train or discard a model?" will naturally arise in this regard. A sophisticated infrastructure for model management will have to be embedded into a software architecture that facilitates the generation of long-term autonomous robot behavior. The Rock framework provides a suitable basis for such an architecture.
- **Open-Source Release** The ARC-OPT framework has been applied to various robotic systems at the DFKI RIC institute (see Figure 6.14). To increase visibility outside RIC, the framework shall be published as open-source software. In particular, the integration of further solvers, robot models and WBC problems can be aided by the robotics community. In return, ARC-OPT facilitates benchmarking different WBC approaches, which can be utilized by other researchers. Furthermore, the community may profit from the various existing WBC implementations and the surrounding learning architecture.



# APPENDIX





---

# Kinematics of Open Chains

---

## A.1 Differential Kinematics

The relationship between task space and configuration space variables of a robot manipulator can be established on position, velocity and acceleration level. On position-level, the equation

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \quad (\text{A.1})$$

relates the joint space coordinates  $\mathbf{q} \in \mathbb{R}^N$  to the task space coordinates  $\mathbf{x} \in \mathbb{R}^M$ , where  $\mathbf{f}$  is a non-linear vector function,  $N$  the number of robot joints and  $M$  the number of task space variables. A typical task is end-effector positioning. When considering both, position and orientation of the end effector, a minimal representation of  $\mathbf{x}$  is  $\mathbf{x} = (x, y, z, \phi, \theta, \psi)^T$ , where  $(x, y, z)^T$  denote the end-effector's position and  $(\phi, \theta, \psi)^T$  its orientation as roll-pitch-yaw Euler angles. If  $N > M$  the robot is said to be kinematically redundant. When differentiating (A.1), we obtain the first-order differential kinematics equation:

$$\dot{\mathbf{x}} = \frac{\delta \mathbf{f}}{\delta \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}_t(\mathbf{q}) \dot{\mathbf{q}} \quad (\text{A.2})$$

where  $\dot{\mathbf{q}}$  is the joint velocity vector,  $\dot{\mathbf{x}}$  the task space velocity vector and  $\mathbf{J}_t \in \mathbb{R}^{M \times N}$  the analytic Jacobian matrix of the robot, also referred to as task Jacobian. It is important to note that, regarding the end-effector orientation, the analytic Jacobian does not relate joint velocities to the angular velocities  $\omega$  of the end effector. Instead, it relates the joint velocities to the rate of change of the parameters characterized by the chosen representation of orientation, in this case the derivative of the chosen roll-pitch-yaw Euler angles. In contrast to that, the geometric Jacobian  $\mathbf{J}$  relates spatial end effector velocity  $\mathbf{v} \in \mathbb{R}^6$  to joint velocities  $\dot{\mathbf{q}}$ :

$$\mathbf{v} = \mathbf{J}(\mathbf{q}) \dot{\mathbf{q}} \quad (\text{A.3})$$

Analytic and geometric Jacobian are related via a simple transform:

$$\mathbf{J}_t = \mathbf{T} \mathbf{J} \quad (\text{A.4})$$

Another common distinction is made regarding whether the Jacobian is expressed in fixed (space) frame coordinates or in moving (body) frame coordinates. The respective Jacobians

are referred to as space Jacobian and body Jacobian. The space Jacobian  $\mathbf{J}_s$  relates the joint velocities to the twist  $\mathbf{v}_s$  expressed in spatial coordinates, e.g., the base frame of a robot. In contrast, the body Jacobian  $\mathbf{J}_b$  relates the joint velocities to the twist  $\mathbf{v}_b$  expressed in body coordinates, e.g., the end-effector frame. Both types are frequently used in modeling inverse kinematics problems and the choice depends on the coordinate frame in which the problem should be expressed.

The second order differential kinematics can be obtained by differentiating (A.3):

$$\dot{\mathbf{v}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} \quad (\text{A.5})$$

where  $\ddot{\mathbf{q}} \in \mathbb{R}^N$  is the robot joint acceleration and  $\dot{\mathbf{v}} \in \mathbb{R}^6$  the spatial acceleration of the end effector.

## A.2 Singularities

The configuration  $\mathbf{q}$  of a robotic manipulator is said to be singular, if the Jacobian matrix  $\mathbf{J}(\mathbf{q})$  is singular, i.e., if it is rank deficient. Note that, since the task Jacobian  $\mathbf{J}_t$  is computed according to (A.4), we have to distinguish representation and kinematic singularities. The former are related to rank deficiencies of  $\mathbf{T}$  and depend on the chosen representation of the end effector orientation. The latter occur when the geometric Jacobian is singular and are related to a loss of mobility of the end effector. In this case there exist end effector velocities that are unfeasible for any joint velocity command.

The singular value decomposition (SVD) is a tool to analyze the singularity of any linear mapping. The SVD of the Jacobian is computed as:

$$\mathbf{J} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad (\text{A.6})$$

where  $\mathbf{U} \in \mathbb{R}^{M \times M}$  and  $\mathbf{V} \in \mathbb{R}^{N \times N}$  are orthonormal matrices containing the left- and right-singular vectors and  $\mathbf{\Sigma} \in \mathbb{R}^{M \times N}$  is a diagonal matrix containing the singular values on its main diagonal. If all singular values  $\sigma_1, \dots, \sigma_M$  are non-zero, the Jacobian is full-rank and the range space of  $\mathbf{J}$  is the entire  $\mathbb{R}^M$ . Close to a singularity at least one singular value approaches zero, which decreases the dimension of the range space of  $\mathbf{J}$ , eliminating a linear combination of task space velocities from the space of feasible velocities.

When approaching a singularity, at some point the required joint space velocities for a given task space velocity will become extremely high. The effect can be analyzed using the manipulability measure, which characterizes the distance from a singularity:

$$\mu = \sqrt{|\mathbf{J}\mathbf{J}^T|} = \prod_{i=1}^M \sigma_i \quad (\text{A.7})$$

Maximizing  $\mu$  will drive the manipulator away from singular configurations.

### A.3 Inverse Differential Kinematics

The inverse differential kinematics problem can be solved on position, velocity, or acceleration level by inverting either (A.1), (A.3) or (A.5). Considering WBC, only the latter two are relevant. If the manipulator is redundant ( $N > M$ ) the general solution of (A.3) or (A.5) can be expressed with the help of the pseudo inverse  $\mathbf{J}^+ \in \mathbb{R}^{N \times M}$  of the Jacobian. The pseudo inverse of  $\mathbf{J}$  is a unique matrix satisfying the Moore-Penrose conditions:

$$\begin{aligned} \mathbf{J}\mathbf{J}^+ &= \mathbf{J} \\ \mathbf{J}^+\mathbf{J} &= \mathbf{J}^+ \\ (\mathbf{J}\mathbf{J}^+)^T &= \mathbf{J}\mathbf{J}^+ \\ (\mathbf{J}^+\mathbf{J})^T &= \mathbf{J}^+\mathbf{J} \end{aligned} \quad (\text{A.8})$$

If  $\mathbf{J}$  is full-rank, its pseudo inverse can be computed as

$$\mathbf{J}^+ = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1} \quad (\text{A.9})$$

Using the pseudo inverse of the Jacobian, the general solution of the first-order inverse differential kinematics can be expressed as:

$$\dot{\mathbf{q}} = \mathbf{J}^+\mathbf{v} + (\mathbf{I} - \mathbf{J}^+\mathbf{J})\dot{\mathbf{q}}_0 \quad (\text{A.10})$$

where  $\dot{\mathbf{q}}_0$  is an arbitrary joint space velocity and the operator  $\mathbf{P} = (\mathbf{I} - \mathbf{J}^+\mathbf{J})$  is the orthogonal projection in the null space of  $\mathbf{J}$ . The null space of  $\mathbf{J}$  is the set of joint space velocities that yield zero task space velocity. The term  $(\mathbf{I} - \mathbf{J}^+\mathbf{J})\dot{\mathbf{q}}_0$  is therefore a null space velocity. By acting on  $\dot{\mathbf{q}}_0$  different joint space velocities can be obtained that result in the same task space velocity. This facilitates the application of arbitrary secondary tasks that can be solved in the null space of the primary task defined by  $\mathbf{v}$ .

The second-order inverse differential kinematics solution can be expressed by solving (A.5):

$$\ddot{\mathbf{q}} = \mathbf{J}^+(\dot{\mathbf{v}} - \dot{\mathbf{J}}\dot{\mathbf{q}}) + (\mathbf{I} - \mathbf{J}^+\mathbf{J})\ddot{\mathbf{q}} \quad (\text{A.11})$$

### A.4 Singularity Robustness

For computational reasons and due to the presence of kinematic singularities, the pseudo inverse of the Jacobian is hardly computed using (A.9) in practice. Multiple methods to provide robustness with respect to singularities exist, one of which is called regularization or damped least-squares technique. Using the singular value decomposition of  $\mathbf{J}$ :

$$\mathbf{J}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T \quad (\text{A.12})$$

the damped pseudo inverse can be computed by approximating the diagonal entries  $s_i^{-1}$  of  $\Sigma^+$  as follows:

$$s_i^{-1} = \frac{s_i}{s_i^2 + \lambda^2}, \quad \forall i \quad (\text{A.13})$$

where  $s_i$  are the singular values of the Jacobian. The damping factor  $\lambda$  can be fixed or computed from the current singular values according to [MK88]:

$$\lambda = \begin{cases} 0 & \text{if } s_m \geq q_{max}^{-1} \\ (2q_{max})^{-1} & \text{if } s_m \leq (2q_{max})^{-1} \\ \sqrt{s_m (q_{max}^{-1} - s_m)} & \text{else} \end{cases} \quad (\text{A.14})$$

here  $s_m$  is the minimum eigenvalue of  $\mathbf{J}$  and  $q_{max}$  is the maximum norm of the joint velocities that the solution should allow, a value that must be selected in advance according to capabilities of the robotic manipulator. Using this automatic adaptation of the damping, the inverse solution will be exact if the robot is far from a singularity and approximate (damped) if it is approaching a singular configuration, in which case the singular values of  $\mathbf{J}$  become small. This solution provides nice properties regarding singularity robustness and damping and can safely steer the robot through singular configurations [MK88].

## Dynamics of Open Chains

Considering the forces and torques that cause the motion of kinematic chains leads to the topic of robot dynamics. The dynamic equations that express this problem are referred to as equations of motion (EOM), which have the form:

$$\boldsymbol{\tau} = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \quad (\text{B.1})$$

where  $\boldsymbol{\tau} \in \mathbb{R}^N$  is the vector of joint forces and torques,  $\mathbf{H} \in \mathbb{R}^{N \times N}$  is the symmetric, positive-definite mass-inertia matrix in joint space and  $\mathbf{h} \in \mathbb{R}^N$  accounts for the effect of centripetal, Coriolis and gravitational forces in joint space. Similarly to forward and inverse kinematics, one can distinguish the forward and inverse dynamics. The forward dynamics determines the robot joint acceleration  $\ddot{\mathbf{q}}$  given the robot state  $\mathbf{q}, \dot{\mathbf{q}}$ , as well as the joint forces and torques  $\boldsymbol{\tau}$ . The inverse problem is about finding the required joint forces and torques  $\boldsymbol{\tau}$  given the robot state  $\mathbf{q}, \dot{\mathbf{q}}$  and the desired joint acceleration  $\ddot{\mathbf{q}}$ . The EOM are usually either derived by using the conceptually elegant, but computationally inefficient Lagrangian formulation or by the Newton-Euler formulation, which is better suited for robots with many dof [LP17]. Equation (B.1) considers the case where the robot moving in free space. If the robot is in contact with the environment the wrenches  $\mathbf{f}_c$  in each contact point have to be considered:

$$\boldsymbol{\tau} = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) + \sum_i \mathbf{J}_{c,i}^T \mathbf{f}_c \quad (\text{B.2})$$

where  $\mathbf{J}_{c,i}$  is the contact Jacobian of the  $i$ -th contact.

The mass-inertia matrix  $\mathbf{H}$  is a generalization of a point mass for rigid body open chains. For complex robots, an accurate derivation of the mass-inertia distribution will be hard to obtain [BGK16]. Therefore, algorithms for dynamic control attempt to avoid direct inversion of  $\mathbf{H}$ , which may be problematic in terms of model uncertainties.

In task space the robot dynamics can be expressed as

$$\mathbf{f} = \boldsymbol{\Lambda}(\mathbf{q})\dot{\mathbf{v}} + \boldsymbol{\eta}(\mathbf{q}, \mathbf{v}) \quad (\text{B.3})$$

where  $\mathbf{f} \in \mathbb{R}^6$  is the end effector wrench,  $\boldsymbol{\Lambda} = \mathbf{J}\mathbf{H}\mathbf{J}^{-1} \in \mathbb{R}^{M \times M}$  is the task space mass-inertia matrix and  $\boldsymbol{\eta} \in \mathbb{R}^M$  is the effect of centripetal, Coriolis and gravitational forces in task space.

The task wrench  $\mathbf{f}_t$  applied at a point in operational space of the robot can be obtained by applying the following joint force and torques:

$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{f}_t \quad (\text{B.4})$$

where  $\mathbf{f}_t$  has to account for the task space dynamics of the robot described by (B.3). For a redundant robot,  $\boldsymbol{\tau}$  is not unique and a secondary task can be introduced:

$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{f}_t + (\mathbf{I} - \bar{\mathbf{J}}\mathbf{J})^T \boldsymbol{\tau}_0 \quad (\text{B.5})$$

where  $\bar{\mathbf{P}} = (\mathbf{I} - \bar{\mathbf{J}}\mathbf{J}) \in \mathbb{R}^{N \times N}$  is the dynamically consistent null space projection of  $\mathbf{J}$  and  $\boldsymbol{\tau}_0 \in \mathbb{R}^N$  is a joint torque vector describing an arbitrary secondary objective. The term  $\bar{\mathbf{J}} = \mathbf{H}^{-1} \mathbf{J}^T \boldsymbol{\Lambda}$  is called dynamically consistent generalized inverse of  $\mathbf{J}$ . The use of this specific inverse results in the task-consistent forces and torques that minimize the kinetic energy. The null space term can be used to achieve arbitrary secondary criteria that utilize the robot's redundancy, such as obstacle, singularity or joint limit avoidance, and energy/torque minimization.

# C

---

## Linear Least Squares and Quadratic Programming

---

**Theorem C.0.1** *Every linear least squares problem can be written as a standard quadratic program.*

*Proof.* Let a linear least squares problem be described as

$$\min_{\mathbf{x}} \quad \frac{1}{2} \|\mathbf{Ax} - \mathbf{b}\|_2$$

This can be transferred to a standard quadratic program as follows:

$$\begin{aligned} \frac{1}{2} \|\mathbf{Ax} - \mathbf{b}\|_2 &= \frac{1}{2} (\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b}) \\ &= \frac{1}{2} (\mathbf{x}^T \mathbf{A}^T \mathbf{Ax} - \mathbf{x}^T \mathbf{A}^T \mathbf{b} - \mathbf{b}^T \mathbf{Ax} + \mathbf{b}^T \mathbf{b}) \\ &= \frac{1}{2} (\mathbf{x}^T \mathbf{A}^T \mathbf{Ax} - 2\mathbf{x}^T \mathbf{A}^T \mathbf{b} + \mathbf{b}^T \mathbf{b}) \end{aligned}$$

Since  $\mathbf{b}$  is assumed fixed, it can be ignored in the optimization problem, which gives:

$$\min_{\mathbf{x}} \quad \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{g}^T \mathbf{x}$$

which is a standard quadratic program with  $\mathbf{Q} = \mathbf{A}^T \mathbf{A}$  and  $\mathbf{g} = -\mathbf{A}^T \mathbf{b}$ . □





---

## Relation between WBC and Optimal Control

---

Optimal control is a branch of mathematical optimization that attempts to find a control law for a dynamical system such that a running cost (e.g., the integral of the position tracking error over time) and a terminal cost (e.g., the final position error) is minimized [Tod06]. If the optimal control policy for a dynamical system is computed once, there is no need for the application of feedback controllers since the policy steers the systems in an optimal manner. In practice the model of the dynamical system will be only approximately known, and a stabilizing feedback control is required to cancel out inevitable modeling errors. The optimal control problem is typically subject to multiple constraints, e.g., maximum torque, battery consumption, workspace limits, etc. In robotics, in particular model predictive control (MPC) is of interest, also known as receding horizon control or online trajectory optimization [Ceb+21]. In MPC the optimal control law is computed for a finite number of time steps ahead. Then, the first value of the trajectory is applied to the robot and the optimization problem is solved again with the current state estimate. MPC has been successfully applied in many technical fields like e.g., chemistry. However, in robotics the timescales of system dynamics are much smaller. Furthermore, many robotic tasks involve making and braking of contact with the environment (e.g., in humanoid walking), which poses certain problems to MPC. Still online MPC has been successfully applied to humanoid robots. For example, Del Prete et al. [Del+14] introduce prioritized optimal control, which combines the advantages of task prioritization known from WBC and optimal control theory. The authors evaluate the approach on a simulated humanoid using a control loop running at  $20ms$  and a time horizon of one second. Tassa et al. [TET12a] introduce a MPC approach that relies on a powerful physics simulation engine [TET12b] and enables the simulated robot to perform complex behaviors like getting up from the ground or recovering from large disturbances. Their MPC framework runs at  $8ms$  with a time horizon of  $500ms$ . Kleff et al. [Kle+21] implement closed-loop non-linear MPC at real-time rates (1 KHz) on a 7 dof industrial manipulator.

In contrast to optimal control or MPC, WBC solves an instantaneous optimization problem by regarding cost and constraints only for the current time step. It has the advantage to perform at higher speeds by offloading parts of the optimization problem into a closed-loop projection-based solution [MS19]. On the negative side, WBC-based solutions are less optimal and more likely to get stuck in local minima. A common workaround that combines the advantages of MPC and WBC is to perform trajectory optimization offline and stabilize the optimal trajectories with a whole-body controller during execution [Ceb+21]. This requires a decent simulation of the full task and environment.

In fact, it has been shown by [Kle+21] that Task Space Inverse Dynamics (TSID) is equivalent to optimal control with collapsed time horizon.



# Results on Optimizing Task Priority Functions

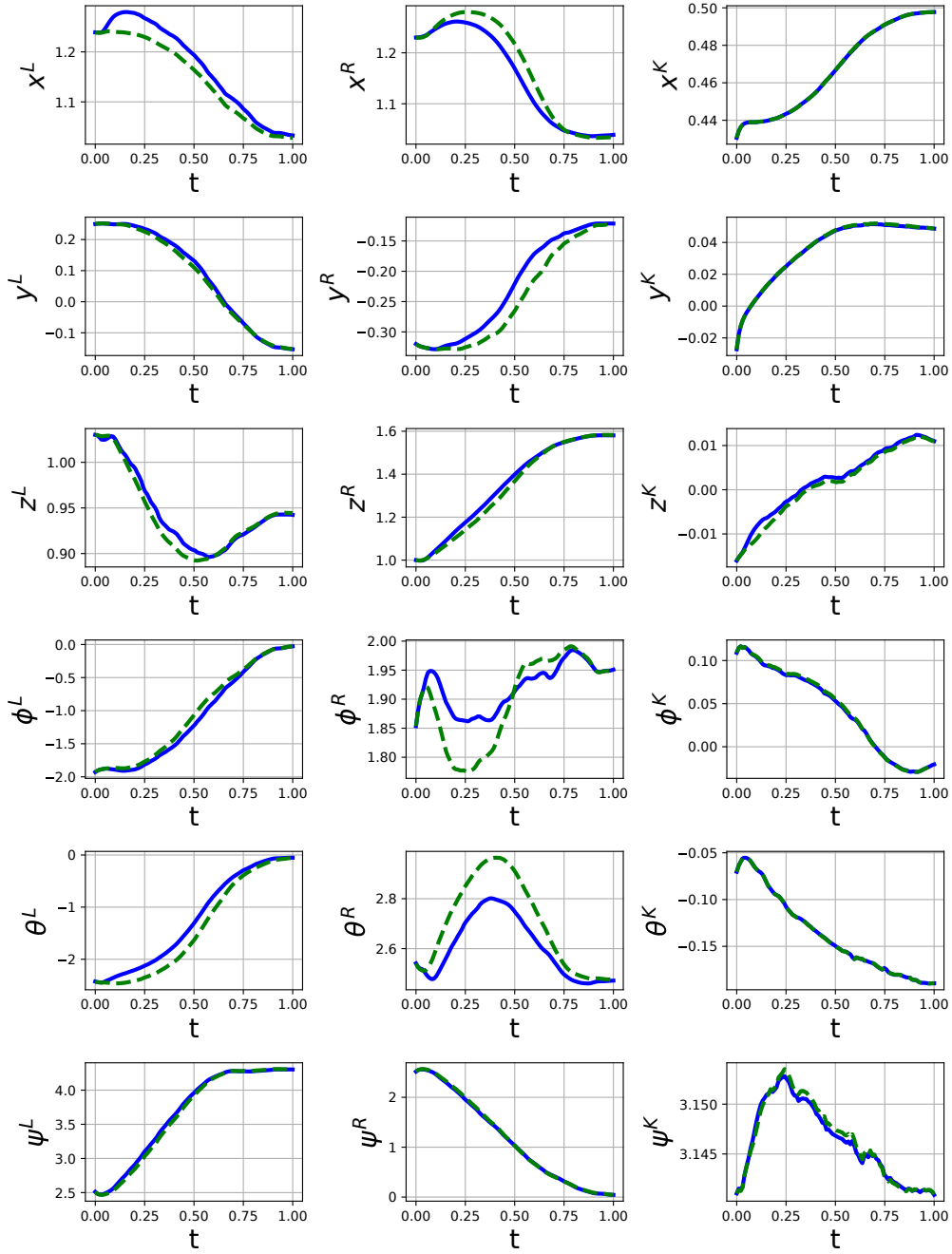


Figure E.1: Resulting trajectories after task priority optimization using cost function  $f_1$ , *Rotate Object* task, KUKA dual-arm robot

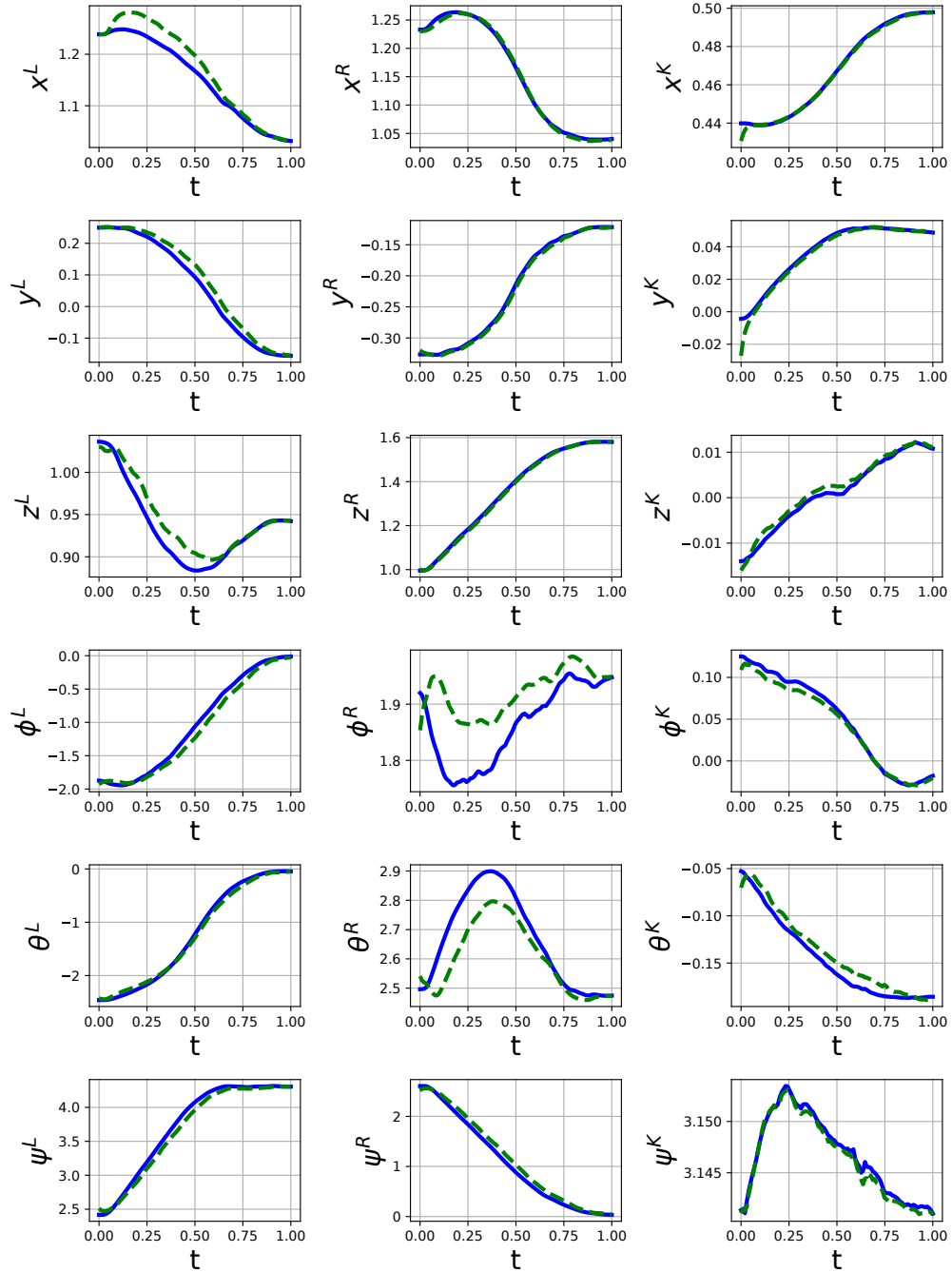


Figure E.2: Resulting trajectories after task priority optimization using cost function  $f_2$ , *Rotate Object* task, KUKA dual-arm robot

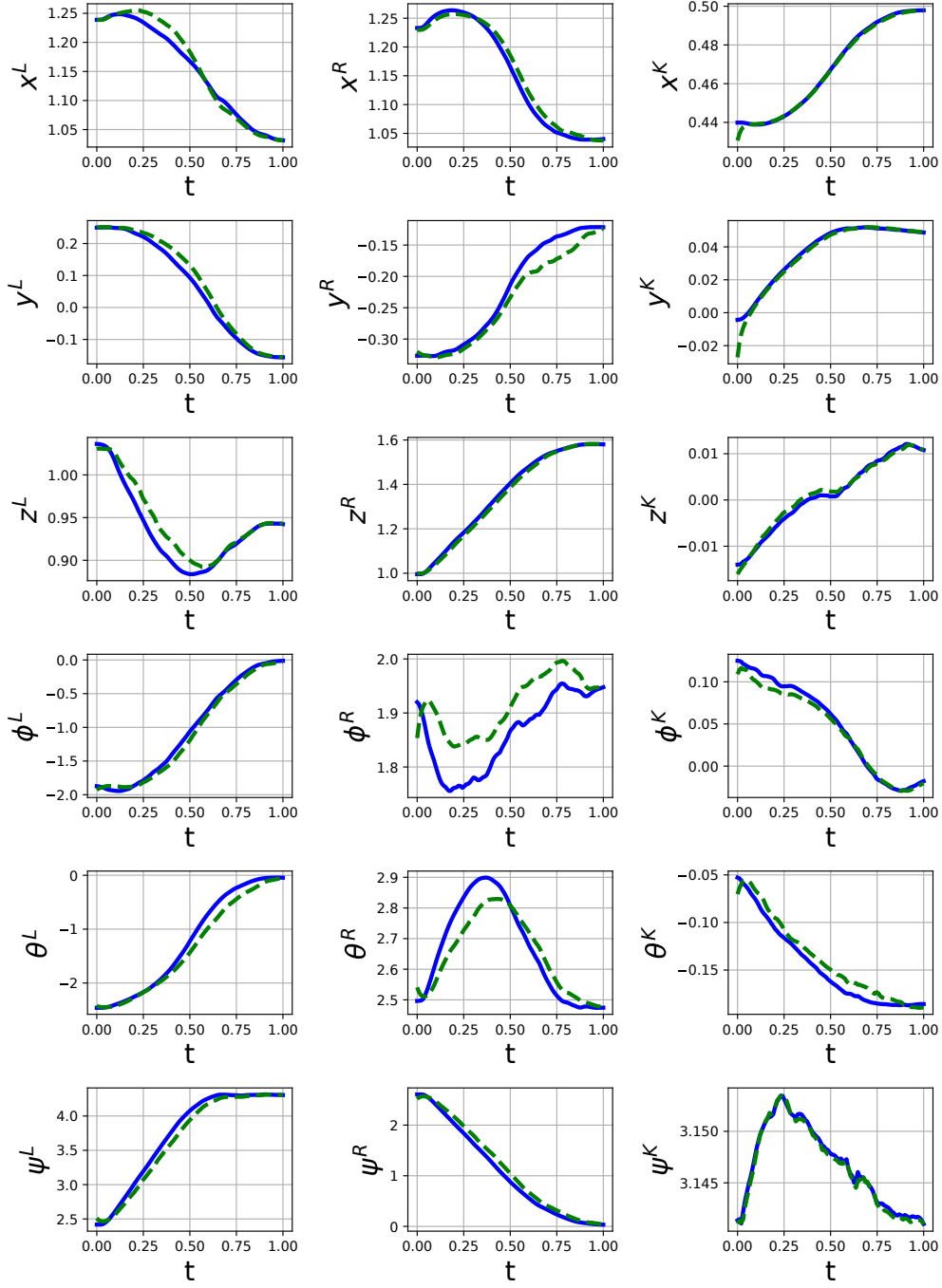


Figure E.3: Resulting trajectories after task priority optimization using cost function  $f_3$ , *Rotate Object* task, KUKA dual-arm robot

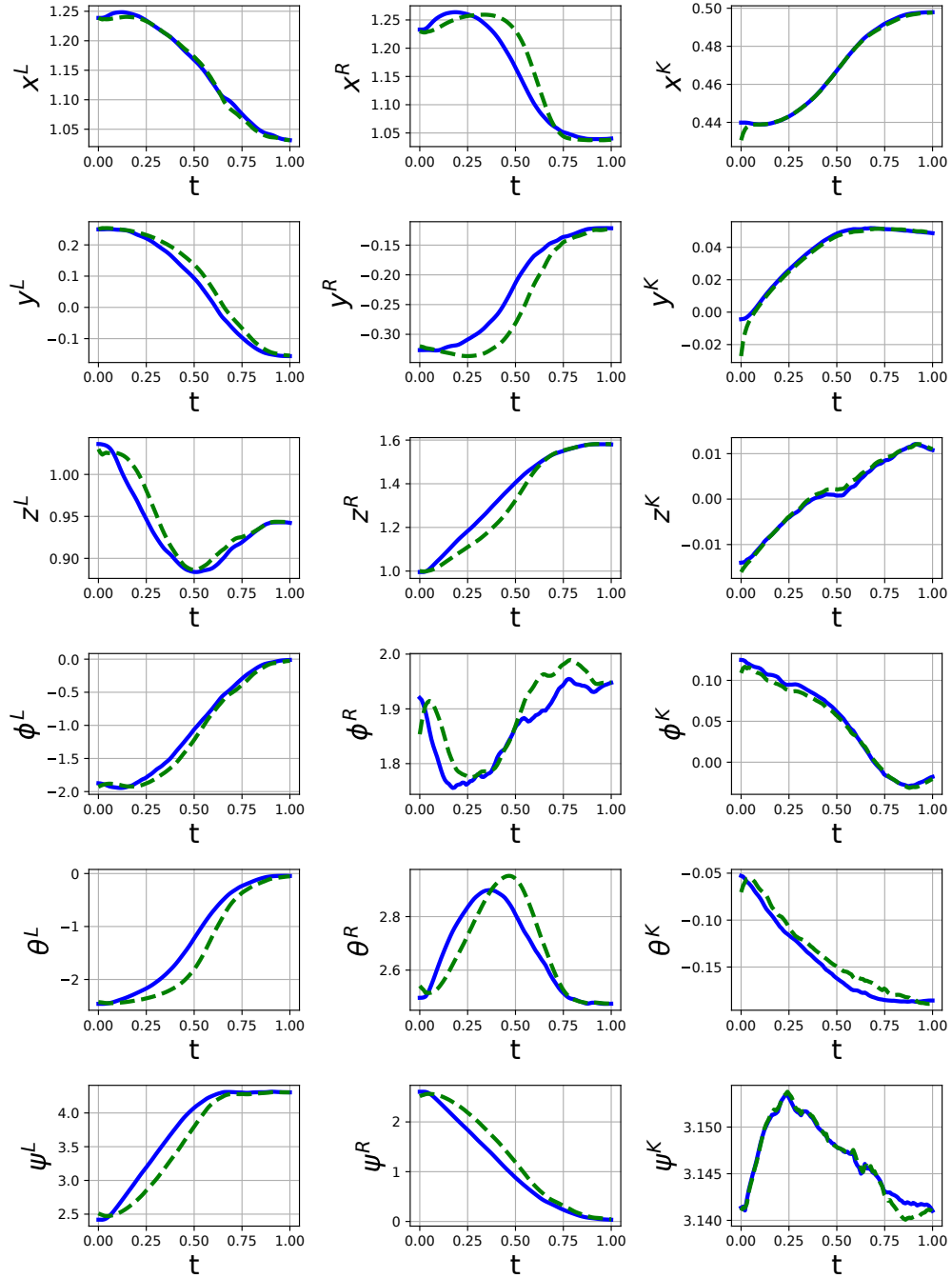


Figure E.4: Resulting trajectories after task priority optimization using cost function  $f_4$ , *Rotate Object* task, KUKA dual-arm robot



# Notations and Symbols

Throughout the document the notations and symbols as shown in the following table will be used. The table only shows the most important variables. Vectors are represented by lowercase bold characters, matrices by uppercase bold characters.

Operators and Attributes			
$\dot{x}$	Time derivative of $x$		
$\hat{x}$	Estimate of $x$		
$\mathbf{A}^{-1}$	Inverse of a matrix $\mathbf{A}$	$\mathbf{J} \in \mathbb{R}^{6 \times N}$	Geometric Jacobian
$\mathbf{A}^+$	Pseudo-inverse of a matrix $\mathbf{A}$	$\mathbf{J}_t \in \mathbb{R}^{6 \times N}$	Task Jacobian
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$	$\mathbf{P} \in \mathbb{R}^{N \times N}$	Null space projection matrix
$[\mathbf{x}]$	Skew symmetric matrix of a vector $\mathbf{x}$ (see e.g., [LP17])	$\boldsymbol{\tau} \in \mathbb{R}^{N_p}$	Robot joint actuation torques
$\text{tr}(\mathbf{A})$	Trace of a square matrix $\mathbf{A}$	$\mathbf{H} \in \mathbb{R}^{N \times N}$	Joint space mass-inertia matrix
<b>Dimensions</b>		$\mathbf{h} \in \mathbb{R}^N$	Vector of bias force/torques in joint space
$N$	Number of robot joints	$\mathbf{f} \in \mathbb{R}^6$	Wrench in Cartesian space
$N_N$	Number of spanning tree robot joints	$\boldsymbol{\Lambda} \in \mathbb{R}^{M \times N}$	Task space mass-inertia matrix
$N_M$	Number of independent robot joints	$\mathbf{R} \in \text{SO}(3)$	Rotation matrix
$N_P$	Number of actuated robot joints	$\theta \in \mathbb{R}$	Rotation angle
$N_q$	Number of decision variables in QP	$\hat{\boldsymbol{\omega}} \in \mathbb{R}^3$	Unit rotation axis
$N_c$	Number of constraints in QP	$\mathbf{u} \in \mathbb{R}^{N_p}$	Robot configuration in actuation space
$M$	Number of constraint/task variables	$\mathbf{W} \in \mathbb{R}^{6 \times 6}$	Diagonal task weight matrix
$D$	Number of demonstrations per context	$\mathbf{S} \in \mathbb{R}^{N_p \times N}$	Actuation matrix
$C$	Number of context variables	$\mathbf{w} \in \mathbb{R}^6$	Task weight vector
$S$	Number of samples per demonstration	$\Delta t$	Sample time
$K$	Number of mixture components	<b>Mixture Models</b>	
$F$	Number of task frames	$p(x)$	Probability distribution of $x$
<b>Robot Control</b>		$\mu$	Mean of a Gaussian
$\mathbf{q} \in \mathbb{R}^N$	Robot configuration in joint space	$\boldsymbol{\Sigma}$	Covariance matrix of a Gaussian
$\mathbf{p} \in \mathbb{R}^3$	Position in Cartesian space	$\sigma^2$	Variance
$\mathbf{x} \in \mathbb{R}^6$	Pose in Cartesian space	$\pi$	Mixing weight in a GMM
$\mathbf{v} \in \mathbb{R}^6$	Twist/spatial velocity in Cartesian space	<b>Data Sets and modeling</b>	
		$\boldsymbol{\kappa} \in \mathbb{R}^C$	Context vector
		$\xi$	Multi-dimensional data set
		$\mathbf{X} \in \mathbb{R}^{D \cdot S \times L}$	Data set with poses
		$\mathbf{V} \in \mathbb{R}^{D \cdot S \times L}$	Data set with twists
		$\mathcal{K} \in \mathbb{R}^{D \cdot S \times C}$	Data set with contexts



---

# List of Figures

---

1.1	Illustrative example on the effect of different task hierarchies. . . . .	2
1.2	Illustrative example on the effect of context changes. Screenshot from video [Mro21d]	3
1.3	Robotic systems used for experimental evaluation in this thesis. . . . .	4
1.4	Overview of the contributions of this thesis. . . . .	7
2.1	Illustrations on Whole-Body Control . . . . .	10
2.2	Model of a floating base robot . . . . .	24
2.3	General idea of PbD, figure inspired by [Bil+08] . . . . .	25
2.4	Using GMM to model a data set obtained from $D = 10$ user demonstrations ( $K = 3$ ). GMR is applied to retrieve a continuous motion (blue dashed line). . . . .	28
3.1	Learning task priority functions: Optimized task priorities yield improved WBC performance compared to manually selected ones, drawing inspired by Modugno et al. [Mod+16b]. . . . .	37
3.2	Knowledge-based approaches parameterize Whole-Body Controllers using sym- bolic reasoning, drawing inspired by Leidner et al. [Lei+16] . . . . .	39
4.1	Overview of the approach on learning context-adaptive task constraints from user demonstrations . . . . .	45
4.2	Modeling of task constraints. . . . .	47
4.3	Overview of the control framework including learning of context-adaptive task constraints . . . . .	51
4.4	Example: Estimated task constraints (x-position) and confidence interval used for predicting soft task priorities. . . . .	53
4.5	Kinesthetic teaching of dual-arm manipulation tasks, screenshots from video [MK21]	55
4.6	Results when reproducing task constraints in previously unseen context: Gray: Mean of demonstrations, Blue Dashed: Left Arm (constraint <i>Base-Left EE</i> ), Green Dashed: Right Arm (constraint <i>Base-Right EE</i> ), Red Dashed: Reproduction in previously unseen context . . . . .	57
4.7	Results on reproduction of task constraints, screenshots from video [MK21] . . . .	58
4.8	Learning curves for the <i>Rotate Object</i> task, x-axis: Number of training samples, y-axis: mean absolute reproduction error for the contexts $R_{11} \dots R_{15}$ and $R_{21} \dots R_{25}$	59
4.9	Estimation of soft task priorities: Temporal, inter-constraint and contextual adapta- tion. <i>Left</i> : Reproduction of task constraints and variance, <i>Right</i> : Estimation of task priorities according to (4.10). . . . .	60
4.10	Comparison of the reproduction error using three different methods for task prioritization: Estimated soft task priorities (approach described in Section 4.5), manually selected soft task priorities (blue) and estimated strict hierarchies as proposed by Silvério et al. [Sil+19] (orange). . . . .	61
4.11	Kinesthetic teaching on the RH5 humanoid. Illustration of task frames and constraints.	64

4.12	Reproduction of task constraints using the ProMP-based approach . . . . .	67
4.13	Results on reproduction of task constraints on the RH5 robot, left arm position. Blue solid: Known contexts, Red dashed: Reproduction in unknown context. . . .	68
4.14	Comparison of the GMM-GMR and the ProMP-based approach for the <i>Rotate Object</i> data set obtained on the KUKA dual-arm robot. . . . .	69
5.1	Overview of the most important sensor processing steps. . . . .	74
5.2	Generation of avoidance motions using KCCD and potential fields. . . . .	75
5.3	Results on task-compliant collision avoidance using WBC. . . . .	78
5.4	Illustration of tasks used in the experimental setup. The dotted white lines indicate which parts of the robot are used for a task. . . . .	79
5.5	GA results. Development of individual fitness measures and the global fitness (left). Development of task weights of the <i>Keep Relative Pose</i> and the <i>Left Arm Position</i> task (right) . . . . .	80
5.6	Comparing the robot behavior for the optimized and manually tuned WBC parameters. —: Reference trajectory, —: Trajectory with manually tuned parameters, ···: Trajectory with optimized parameters . . . . .	82
5.7	Trajectory tracking and collision avoidance with a human entering the workspace of the robot, screenshots from video [Mro21a]. . . . .	83
5.8	Resulting soft task priorities for the three tasks after optimization with CMA-ES .	87
5.9	Result on task priority optimization. . . . .	88
6.1	Overview of ARC-OPT: Components and their interaction . . . . .	92
6.2	Class diagram showing the key components of ARC-OPT . . . . .	95
6.3	Rock integration of the ARC-OPT library exemplified for the RH5 humanoid. . .	96
6.4	WBC configuration example for the RH5 robot. . . . .	97
6.5	PbD pipeline and black-box optimization in ARC-OPT . . . . .	98
6.6	RH5 ankle mechanism, implementing box constraints for actuator positions. Image credits [Kum19]. . . . .	101
6.7	RH5 elbow mechanism, box constraints for actuator velocity/torque. . . . .	102
6.8	Different sub-spaces for modeling series-parallel hybrid robots. Image credits [Kum19] . . . . .	103
6.9	Comparison of squatting motion on RH5 using WBC with abstract/tree-type model and series-parallel hybrid model. . . . .	105
6.10	Squatting motions on the RH5 humanoid. . . . .	105
6.11	Comparison of boxing motions on RH5v2 using WBC with abstract/tree-type model and series-parallel hybrid model. . . . .	107
6.12	Executing boxing motions on RH5v2, screenshots from video [Mro21c]. . . . .	108
6.13	Comparison of serial vs. hybrid model regarding the computation time for a complete cycle (model update, scene update, QP solving) for different robot models and scenes. . . . .	110
6.14	Robotic systems at DFKI RIC that the ARC-OPT framework has been applied to. Image Credits: * Annemarie Popp, ** Thomas Frank, *** Alexander Dettmann (all DFKI)	111
E.1	Resulting trajectories after task priority optimization using cost function $f_1$ , <i>Rotate Object</i> task, KUKA dual-arm robot . . . . .	129
E.2	Resulting trajectories after task priority optimization using cost function $f_2$ , <i>Rotate Object</i> task, KUKA dual-arm robot . . . . .	130

E.3	Resulting trajectories after task priority optimization using cost function $f_3$ , <i>Rotate</i> <i>Object</i> task, KUKA dual-arm robot . . . . .	131
E.4	Resulting trajectories after task priority optimization using cost function $f_4$ , <i>Rotate</i> <i>Object</i> task, KUKA dual-arm robot . . . . .	132



---

# List of Tables

---

1.1	Task hierarchies used in Example 1. . . . .	2
2.1	Overview on existing WBC and redundancy resolution approaches . . . . .	14
2.2	General advantages and disadvantages of closed-form and optimization-based WBC approaches . . . . .	15
3.1	Overview of the state of the art on automatic derivation of task constraints . . . .	34
3.2	Comparison of the main approach for learning task constraints followed in this thesis with the state of the art. . . . .	41
4.1	Example context variables . . . . .	49
4.2	Contexts and context variables used for experimental evaluation, OS - Object Size, CW - Clockwise rotation, LA/RA - Left Arm/Right Arm, AT - Allow Tilt . . . . .	56
4.3	Tasks and context variables used for experimental evaluation on RH5, RA - Rotation angle, WA - Whiteboard Angle, CW - Clockwise, AW - Anticlockwise, HO - Horizontal, LA/RA - Left Arm/Right Arm, TH - Table Height . . . . .	65
5.1	Optimization results: Optimized and manually tuned parameter set. Individual fitness measures $f^L$ , $f^K$ , $f^A$ and global fitness $f$ for the respective parameter sets	81
6.1	Overview of the most popular open-source software frameworks for WBC . . . . .	90
6.2	Overview on the implemented controllers in ARC-OPT. . . . .	93
6.3	Overview of the available scenes in ARC-OPT . . . . .	94
6.4	Complexity of different robots models. . . . .	108





---

# Glossary

---

## **ARC-OPT**

Adaptive Robot Control using Optimization. x, 6, 7, 89–100, 102, 104, 106, 108, 110–113, 115, 136, 139

## **BIC**

Bayesian Information Criterion. 27

## **CMA-ES**

Covariance Matrix Adaptation Evolution Strategy. 37, 38, 85–88, 99, 136

## **CNN**

convolutional neural network. 31

## **CoM**

Center of Mass. 64, 66, 96, 105, 106, 109

## **DL**

Deep Learning. 31

## **DMP**

Dynamic Movement Primitive. 29–31, 38, 40, 63

## **dof**

degrees of freedom. 1, 3, 5, 6, 11, 13, 15–17, 23, 25, 32, 34, 35, 37, 40, 43–45, 47, 70, 79, 80, 83, 84, 86, 87, 93, 95, 100, 101, 104, 109, 123, 127

## **DPGMM**

Dirichlet Process Gaussian Mixture Model. 51, 54, 56–58

## **EM**

Expectation Maximization. 27, 52

## **EOM**

equations of motion. 12, 20, 24, 94, 104, 123

## **GA**

genetic algorithm. 78–81, 136

## **GMM**

Gaussian Mixture Model. 27–29, 31, 35, 45, 51, 54, 63, 66, 68–70, 99, 113, 135, 136

## **GMR**

Gaussian Mixture Regression. 28, 29, 31, 35, 43, 46, 53, 54, 56, 61, 63, 66, 68, 69, 99, 113, 135, 136

## **GPR**

Gaussian Process Regression. 31, 53, 99

## **HLS**

Hierarchical Least Squares. 94, 95

**HMM**

Hidden Markov Model. 31

**HyRoDyn**

Hybrid Robot Dynamics. 94, 97, 101, 102

**iTaSC**

instantaneous Task Specification using Constraints. 14, 16, 36, 90, 91, 94

**KCCD**

Kinematic Continuous Collision Detection. 72–75, 136

**KDL**

Orocos Kinematics and Dynamics Library. 93, 94

**LWPR**

Locally Weighted Projection Regression. 53

**MLP**

Multi-Layer Perceptron. 99

**MPC**

model predictive control. 127

**PbD**

programming by demonstration. ix, x, 4, 6, 9, 25–27, 29, 31–36, 41, 43–45, 54, 69, 70, 84, 87, 98, 99, 113, 114, 135, 136

**ProMP**

Probabilistic Movement Primitive. 30, 43, 63, 64, 66–69, 99, 113, 136

**QP**

quadratic program. 1, 18, 47, 64, 76, 84, 89, 90, 92–96, 103, 104, 107, 109, 110, 136

**RBF**

radial basis function. 35, 37, 63, 68, 85, 86

**RF**

Reinforcement Learning. ix, 33, 34, 37, 38, 40, 41

**RFR**

Random Forest Regression. 36

**Rock**

Robot Construction Kit. 91, 95, 115

**ROS**

Robot Operating System. 95

**RTT**

Real-Time Toolkit. 96

**SoT**

Stack of Tasks. 90

**SVD**

singular value decomposition. 120

**TP-GMM**

Task-Parameterized Gaussian Mixture Model. 29, 35, 36

**TSID**

Task Space Inverse Dynamics. 14, 20, 89, 90, 127

**URDF**

Unified Robot Description Format. 55, 93, 94, 97, 100

**WBC**

Whole-Body Control. 1–6, 9–24, 30–33, 35–41, 43–47, 50, 53–55, 64, 69–74, 76–79, 81–85, 88–92, 94–104, 106, 107, 109, 111, 113–115, 121, 127, 135, 136, 139

**YARP**

Yet Another Robot Platform. 95

**ZMP**

Zero Moment Point. 37, 84



---

# Bibliography

---

- [AD14] Erwin Aertbeliën and Joris De Schutter. ‘eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs’. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 1540–1546. DOI: [10.1109/IRoS.2014.6942760](https://doi.org/10.1109/IRoS.2014.6942760).
- [AG21] KUKA AG. *LBR iiwa*. 2021. URL: <https://www.kuka.com/en-gb/products/robotics-systems/industrial-robots/lbr-iiwa> (visited on 05/20/2021).
- [AL15] S. i. An and D. Lee. ‘Prioritized Inverse Kinematics with Multiple Task Definitions’. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 1423–1430. DOI: [10.1109/ICRA.2015.7139376](https://doi.org/10.1109/ICRA.2015.7139376).
- [Arm+17] L. Armesto, J. Bosga, V. Ivan, and S. Vijayakumar. ‘Efficient learning of constraints and generic null space policies’. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 1520–1526. DOI: [10.1109/ICRA.2017.7989181](https://doi.org/10.1109/ICRA.2017.7989181).
- [Arm+18] Leopoldo Armesto, João Moura, Vladimir Ivan, Mustafa Suphi Erden, Antonio Sala, and Sethu Vijayakumar. ‘Constraint-aware learning of policies by demonstration’. In: *The International Journal of Robotics Research* 37.13-14 (2018), pp. 1673–1689. DOI: [10.1177/0278364918784354](https://doi.org/10.1177/0278364918784354).
- [Bar+16] Sebastian Bartsch, Marc Manz, Peter Kampmann, Alexander Dettmann, Hendrik Hanff, Malte Langosz, Kai von Szadkowski, Jens Hilljegerdes, Marc Simnofske, Philipp Kloss, Manuel Meder, and Frank Kirchner. ‘Development and Control of the Multi-Legged Robot MANTIS’. In: *Proceedings of ISR 2016: 47th International Symposium on Robotics*. 2016, pp. 1–8.
- [BAS14] G. Borghesan, E. Aertbeliën, and J. De Schutter. ‘Constraint- and synergy-based specification of manipulation tasks’. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. May 2014, pp. 397–402. DOI: [10.1109/ICRA.2014.6906887](https://doi.org/10.1109/ICRA.2014.6906887).
- [BB04] Paolo Baerlocher and Ronan Boulic. ‘An inverse kinematics architecture enforcing an arbitrary number of strict priority levels’. In: *Visual Computer* 20.6 (2004), pp. 402–417. DOI: [10.1007/s00371-004-0244-4](https://doi.org/10.1007/s00371-004-0244-4).
- [BBD03] Johan Baeten, Herman Bruyninckx, and Joris De Schutter. ‘Integrated Vision/Force Robotic Servoing in the Task Frame Formalism’. In: *The International Journal of Robotics Research* 22.10-11 (2003), pp. 941–954. DOI: [10.1177/027836490302210010](https://doi.org/10.1177/027836490302210010).
- [BD14] Gianni Borghesan and Joris De Schutter. ‘Constraint-based specification of hybrid position-impedance-force tasks’. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2290–2296. DOI: [10.1109/ICRA.2014.6907176](https://doi.org/10.1109/ICRA.2014.6907176).

- [Bel+18] C. Dario Bellicoso, Fabian Jenelten, Christian Gehring, and Marco Hutter. ‘Dynamic Locomotion Through Online Nonlinear Motion Optimization for Quadrupedal Robots’. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2261–2268. doi: [10.1109/LRA.2018.2794620](https://doi.org/10.1109/LRA.2018.2794620).
- [Ben80] Adi Ben-Israel. ‘Generalized Inverses of Matrices and Their Applications’. In: *Extremal Methods and Systems Analysis*. Ed. by Anthony V. Fiacco and Kenneth O. Kortanek. Berlin, Heidelberg: Springer Berlin Heidelberg, 1980, pp. 154–186.
- [BGK16] Vinzenz Bargsten, Jose de Gea Fernandez, and Yohannes Kassahun. ‘Experimental Robot Inverse Dynamics Identification Using Classical and Machine Learning Techniques’. In: *Proceedings of ISR 2016: 47st International Symposium on Robotics*. 2016, pp. 1–6.
- [BHB84] John Baillieul, John Hollerbach, and Roger Brockett. ‘Programming and control of kinematically redundant manipulators’. In: *The 23rd IEEE Conference on Decision and Control*. 1984, pp. 768–774. doi: [10.1109/CDC.1984.272110](https://doi.org/10.1109/CDC.1984.272110).
- [Bil+08] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. ‘Robot Programming by Demonstration’. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1371–1394. doi: [10.1007/978-3-540-30301-5\\_60](https://doi.org/10.1007/978-3-540-30301-5_60).
- [Bis06] Christopher M. Bishop. ‘Pattern recognition and machine learning’. In: *Machine learning* 128.9 (2006), pp. 523–558.
- [BK02] Oliver Brock and Oussama Khatib. ‘Elastic Strips: A Framework for Motion Generation in Human Environments’. In: *The International Journal of Robotics Research* 21.12 (2002), pp. 1031–1052. doi: [10.1177/0278364902021012002](https://doi.org/10.1177/0278364902021012002).
- [BKB13] Georg Bartels, Ingo Kresse, and Michael Beetz. ‘Constraint-based movement representation grounded in geometric features’. In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2013, pp. 547–554. doi: [10.1109/HUMANOIDS.2013.7030027](https://doi.org/10.1109/HUMANOIDS.2013.7030027).
- [BKM17] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. ‘Variational Inference: A Review for Statisticians’. In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877. doi: [10.1080/01621459.2017.1285773](https://doi.org/10.1080/01621459.2017.1285773).
- [Blo+16] Domenico D. Bloisi, Daniele Nardi, Francesco Riccio, and Francesco Trapani. ‘Context in Robotics and Information Fusion’. In: *Context-Enhanced Information Fusion: Boosting Real-World Performance with Domain Knowledge*. Ed. by Lauro Snidaro, Jesús García, James Llinas, and Erik Blasch. Cham: Springer International Publishing, 2016, pp. 675–699. doi: [10.1007/978-3-319-28971-7\\_25](https://doi.org/10.1007/978-3-319-28971-7_25).
- [BMT10] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. ‘CRAM — A Cognitive Robot Abstract Machine for everyday manipulation in human environments’. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 1012–1017. doi: [10.1109/IR0S.2010.5650146](https://doi.org/10.1109/IR0S.2010.5650146).
- [Boh+11] Jonathan Bohren, Radu Bogdan Rusu, E. Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. ‘Towards autonomous robotic butlers: Lessons learned with the PR2’. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5568–5575. doi: [10.1109/ICRA.2011.5980058](https://doi.org/10.1109/ICRA.2011.5980058).



- [Bou+22] Melya Boukheddimi, Shivesh Kumar, Heiner Peters, Dennis Mronga, Rohan Budhiraja, and Frank Kirchner. 'Introducing RH5V2: A Powerful Humanoid Upper Body Design for Dynamic Movements'. 2022 IEEE International Conference on Robotics and Automation (ICRA), Accepted for publication. 2022.
- [BSK03] H. Bruyninckx, P. Soetens, and B. Koninckx. 'The real-time motion control core of the Orocos project'. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. 2003, 2766–2771 vol.2. doi: [10.1109/ROBOT.2003.1242011](https://doi.org/10.1109/ROBOT.2003.1242011).
- [Buo21] Gabriele Buondonno. *Eiquadprog*. 2021. URL: <https://github.com/stack-of-tasks/eiquadprog> (visited on 08/02/2021).
- [Bus+17] Baptiste Busch, Guilherme Maeda, Yoan Mollard, Marie Demangeat, and Manuel Lopes. 'Postural optimization for an ergonomic human-robot interaction'. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2778–2785. doi: [10.1109/IROS.2017.8206107](https://doi.org/10.1109/IROS.2017.8206107).
- [Cac+17] R. Caccavale, M. Saveriano, G. A. Fontanelli, F. Ficuciello, D. Lee, and A. Finzi. 'Imitation learning and attentional supervision of dual-arm structured tasks'. In: *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. 2017, pp. 66–71. doi: [10.1109/DEVLRN.2017.8329789](https://doi.org/10.1109/DEVLRN.2017.8329789).
- [Cal+12] Sylvain Calinon, Zhibin Li, Tohid Alizadeh, Nikos G. Tsagarakis, and Darwin G. Caldwell. 'Statistical dynamical systems for skills acquisition in humanoids'. In: *IEEE-RAS International Conference on Humanoid Robots* (2012), pp. 323–329. doi: [10.1109/HUMANOIDS.2012.6651539](https://doi.org/10.1109/HUMANOIDS.2012.6651539).
- [Cal16] Sylvain Calinon. 'A tutorial on task-parameterized movement learning and retrieval'. In: *Intelligent Service Robotics* 9.1 (2016), pp. 1–29. doi: [10.1007/s11370-015-0187-9](https://doi.org/10.1007/s11370-015-0187-9).
- [Car+19] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. 'The Pinocchio C++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives'. In: *2019 IEEE/SICE International Symposium on System Integration (SII)*. 2019, pp. 614–619. doi: [10.1109/SII.2019.8700380](https://doi.org/10.1109/SII.2019.8700380).
- [CB07] Sylvain Calinon and Aude Billard. 'Active Teaching in Robot Programming by Demonstration'. In: *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*. 2007, pp. 702–707. doi: [10.1109/ROMAN.2007.4415177](https://doi.org/10.1109/ROMAN.2007.4415177).
- [CB08] Sylvain Calinon and Aude Billard. 'A probabilistic programming by demonstration framework handling constraints in joint space and task space'. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (2008), pp. 367–372. doi: [10.1109/IROS.2008.4650593](https://doi.org/10.1109/IROS.2008.4650593).
- [CB09] Sylvain Calinon and Aude Billard. 'Statistical Learning by Imitation of Competing Constraints in Joint Space and Task Space'. In: *Advanced Robotics* 23.15 (2009), pp. 2059–2076. doi: [10.1163/016918609X12529294461843](https://doi.org/10.1163/016918609X12529294461843).

- [Ceb+21] Oguzhan Cebe, Carlo Tiseo, Guiyang Xin, Hsiu-chin Lin, Joshua Smith, and Michael Mistry. 'Online Dynamic Trajectory Optimization and Control for a Quadruped Robot'. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 12773–12779. doi: [10.1109/ICRA48506.2021.9561592](https://doi.org/10.1109/ICRA48506.2021.9561592).
- [CGB07] Sylvain Calinon, Florent Guenter, and Aude Billard. 'On learning, representing, and generalizing a task in a humanoid robot'. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 37.2 (2007), pp. 286–298. doi: [10.1109/TSMCB.2006.886952](https://doi.org/10.1109/TSMCB.2006.886952).
- [Cha+18] Marie Charbonneau, Valerio Modugno, Francesco Nori, Giuseppe Oriolo, Daniele Pucci, and Serena Ivaldi. 'Learning Robust Task Priorities of QP-Based Whole-Body Torque-Controllers'. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. 2018, pp. 1–9. doi: [10.1109/HUMANOIDS.2018.8624995](https://doi.org/10.1109/HUMANOIDS.2018.8624995).
- [CKT19] Stephane Caron, Abderrahmane Kheddar, and Olivier Tempier. 'Stair climbing stabilization of the HRP-4 humanoid robot using whole-body admittance control'. In: *Proceedings - IEEE International Conference on Robotics and Automation* 2019-May (2019), pp. 277–283. doi: [10.1109/ICRA.2019.8794348](https://doi.org/10.1109/ICRA.2019.8794348).
- [CM00] F. Chaumette and E. Marchand. 'A new redundancy-based iterative scheme for avoiding joint limits. Application to visual servoing'. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 2. 2000, 1720–1725 vol.2. doi: [10.1109/ROBOT.2000.844844](https://doi.org/10.1109/ROBOT.2000.844844).
- [Col+07] C. Collette, A. Micaelli, C. Andriot, and P. Lemerle. 'Dynamic balance control of humanoids for multiple grasps and non coplanar frictional contacts'. In: *2007 7th IEEE-RAS International Conference on Humanoid Robots*. Nov. 2007, pp. 81–88. doi: [10.1109/ICHR.2007.4813852](https://doi.org/10.1109/ICHR.2007.4813852).
- [DBD13] Wilm Decre, Herman Bruyninckx, and Joris De Schutter. 'Extending the iTaSC Constraint-based Robot Task Specification Framework to Time-Independent Trajectories and User-Configurable Task Horizons'. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. 2013, pp. 1941–1948. doi: [10.1109/ICRA.2013.6630835](https://doi.org/10.1109/ICRA.2013.6630835).
- [Dec+09] Wilm Decre, Ruben Smits, Herman Bruyninckx, and Joris De Schutter. 'Extending iTaSC to support inequality constraints and non-instantaneous task specification'. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 964–971. doi: [10.1109/ROBOT.2009.5152477](https://doi.org/10.1109/ROBOT.2009.5152477).
- [Del+14] Andrea Del Prete, Francesco Romano, Lorenzo Natale, Giorgio Metta, Giulio Sandini, and Francesco Nori. 'Prioritized optimal control'. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2014), pp. 2540–2545. doi: [10.1109/ICRA.2014.6907214](https://doi.org/10.1109/ICRA.2014.6907214).
- [Del+16] Andrea Del Prete, Nicolas Mansard, Oscar E. Ramos, Olivier Stasse, and Francesco Nori. 'Implementing Torque Control with High-Ratio Gear Boxes and Without Joint-Torque Sensors'. In: *International Journal of Humanoid Robotics* 13.01 (2016). Software Documentation: <https://github.com/stack-of-tasks/tsid>. doi: [10.1142/S0219843615500449](https://doi.org/10.1142/S0219843615500449).

- [DF12] Alessandro De Luca and Fabrizio Flacco. ‘Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration’. In: *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. 2012, pp. 288–295. doi: [10.1109/BioRob.2012.6290917](https://doi.org/10.1109/BioRob.2012.6290917).
- [Die+12a] A Dietrich, T. Wimbock, A Albu-Schaffer, and G. Hirzinger. ‘Integration of Reactive, Torque-Based Self-Collision Avoidance Into a Task Hierarchy’. In: *IEEE Transactions on Robotics* 28.6 (Dec. 2012), pp. 1278–1293. doi: [10.1109/TR0.2012.2208667](https://doi.org/10.1109/TR0.2012.2208667).
- [Die+12b] Alexander Dietrich, Thomas Wimbock, Alin Albu-Schaffer, and Gerd Hirzinger. ‘Reactive Whole-Body Control: Dynamic Mobile Manipulation Using a Large Number of Actuated Degrees of Freedom’. In: *IEEE Robotics Automation Magazine* 19.2 (2012), pp. 20–33. doi: [10.1109/MRA.2012.2191432](https://doi.org/10.1109/MRA.2012.2191432).
- [DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. ‘Maximum Likelihood from Incomplete Data Via the EM Algorithm’. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. doi: <https://doi.org/10.1111/j.2517-6161.1977.tb01600.x>.
- [DRS15] N. Dehio, R. F. Reinhart, and J. J. Steil. ‘Multiple task optimization with a mixture of controllers for motion generation’. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015, pp. 6416–6421. doi: [10.1109/IROS.2015.7354294](https://doi.org/10.1109/IROS.2015.7354294).
- [DRS16] N. Dehio, R. F. Reinhart, and J. J. Steil. ‘Continuous task-priority rearrangement during motion execution with a mixture of torque controllers’. In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. Nov. 2016, pp. 264–270. doi: [10.1109/HUMANOIDS.2016.7803287](https://doi.org/10.1109/HUMANOIDS.2016.7803287).
- [DWE14] Dimitar Dimitrov, Pierre-Brice Wieber, and Adrien Escande. ‘Multi-Objective Control of Robots’. In: *Journal of the Robotics Society of Japan* 32.6 (July 2014), pp. 512–518. doi: [10.7210/jrsj.32.512](https://doi.org/10.7210/jrsj.32.512).
- [ECR92] B. Espiau, F. Chaumette, and P. Rives. ‘A new approach to visual servoing in robotics’. In: *IEEE Transactions on Robotics and Automation* 8.3 (1992), pp. 313–326. doi: [10.1109/70.143350](https://doi.org/10.1109/70.143350).
- [EMW14] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. ‘Hierarchical quadratic programming: Fast online humanoid-robot motion generation’. In: *The International Journal of Robotics Research* 33.7 (June 2014), pp. 1006–1028. doi: [10.1177/0278364914521306](https://doi.org/10.1177/0278364914521306).
- [Ess+21] Julian Esser, Shivesh Kumar, Heiner Peters, Vinzenz Bargsten, Jose de Gea Fernandez, Carlos Mastalli, Olivier Stasse, and Frank Kirchner. ‘Design, analysis and control of the series-parallel hybrid RH5 humanoid robot’. In: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. 2021, pp. 400–407. doi: [10.1109/HUMANOIDS47582.2021.9555770](https://doi.org/10.1109/HUMANOIDS47582.2021.9555770).
- [Fab+19] Alexander Fabisch, Christoph Petzoldt, Marc Otto, and Frank Kirchner. ‘A Survey of Behavior Learning Applications in Robotics - State of the Art and Perspectives’. In: *CoRR abs/1906.01868* (2019).
- [Fab21] Alexander Fabisch. ‘gmr: Gaussian Mixture Regression’. In: *Journal of Open Source Software* 6.62 (2021), p. 3054. doi: [10.21105/joss.03054](https://doi.org/10.21105/joss.03054).

- [FBB16] Z. Fang, G. Bartels, and M. Beetz. ‘Learning models for constraint-based motion parameterization from interactive physics-based simulation’. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016, pp. 4005–4012. doi: [10.1109/IROS.2016.7759590](https://doi.org/10.1109/IROS.2016.7759590).
- [FBD08] H. J. Ferreau, H. G. Bock, and M. Diehl. ‘An online active set strategy to overcome the limitations of explicit MPC’. In: *International Journal of Robust and Nonlinear Control* 18.8 (2008), pp. 816–830. doi: [10.1002/rnc.1251](https://doi.org/10.1002/rnc.1251).
- [FDK12] F. Flacco, A. De Luca, and O. Khatib. ‘Motion control of redundant robots under joint constraints: Saturation in the Null Space’. In: *2012 IEEE International Conference on Robotics and Automation (ICRA)*. May 2012, pp. 285–292. doi: [10.1109/ICRA.2012.6225376](https://doi.org/10.1109/ICRA.2012.6225376).
- [Fea14] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer US, 2014.
- [Fen+15] Siyuan Feng, Eric Whitman, X. Xinjilefu, and Christopher G. Atkeson. ‘Optimization based full body control for the atlas robot’. In: *IEEE-RAS International Conference on Humanoid Robots* (2015), pp. 120–127. doi: [10.1109/HUMANOIDS.2014.7041347](https://doi.org/10.1109/HUMANOIDS.2014.7041347).
- [Fer+14] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. ‘qpOASES: A parametric active-set algorithm for quadratic programming’. In: *Mathematical Programming Computation* 6.4 (2014), pp. 327–363. doi: [10.1007/s12532-014-0071-1](https://doi.org/10.1007/s12532-014-0071-1).
- [Fla+12] Fabrizio Flacco, Torsten Kröger, Alessandro De Luca, and Oussama Khatib. ‘A depth space approach to human-robot collision avoidance’. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 338–345. doi: [10.1109/ICRA.2012.6225245](https://doi.org/10.1109/ICRA.2012.6225245).
- [FM14] Alexander Fabisch and Jan Hendrik Metzen. ‘Active Contextual Policy Search’. In: *Journal of Machine Learning Research* 15.1 (Jan. 2014), pp. 3371–3399.
- [For+12] Felix-Antoine Fortin, Francois-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagne. ‘DEAP: Evolutionary Algorithms Made Easy’. In: *Journal of Machine Learning Research* 13 (July 2012), pp. 2171–2175.
- [Fou21a] Open Robotics Foundation. *The Robot Operating System (ROS)*. 2021. URL: <https://www.ros.org/> (visited on 08/02/2021).
- [Fou21b] Open Robotics Foundation. *The ROS transmission interface*. 2021. URL: [http://wiki.ros.org/transmission\\_interface](http://wiki.ros.org/transmission_interface) (visited on 08/11/2021).
- [Fou21c] Open Source Robotics Foundation. *GAZEBO - Robot Simulation Made Easy*. 2021. URL: <http://gazebo.org> (visited on 05/20/2021).
- [Fuc+09] M. Fuchs, Ch. Borst, P. Robuffo Giordano, A. Baumann, E. Kraemer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze, R. Burger, F. Schmidt, T. Wimboeck, and G. Hirzinger. ‘Rollin’ Justin - Design considerations and realization of a mobile platform for a humanoid upper body’. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 4131–4137. doi: [10.1109/ROBOT.2009.5152464](https://doi.org/10.1109/ROBOT.2009.5152464).

- [Gea+17a] José de Gea Fernández, Dennis Mrona, Martin Günther, Tobias Knobloch, Malte Wirkus, Martin Schröer, Mathias Trampler, Stefan Stiene, Elsa Kirchner, Vinzenz Bargsten, Timo Bänziger, Johannes Teiwes, Thomas Krüger, and Frank Kirchner. ‘Multimodal sensor-based whole-body control for human–robot collaboration in industrial settings’. In: *Robotics and Autonomous Systems* 94 (2017), pp. 102–119. doi: <https://doi.org/10.1016/j.robot.2017.04.007>.
- [Gea+17b] José de Gea Fernández, Dennis Mrona, Martin Günther, Malte Wirkus, Martin Schröer, Stefan Stiene, Elsa Kirchner, Vinzenz Bargsten, Timo Bänziger, Johannes Teiwes, Thomas Krüger, and Frank Kirchner. ‘iMRK: Demonstrator for Intelligent and Intuitive Human–Robot Collaboration in Industrial Manufacturing’. In: *KI - Kunstliche Intelligenz* 31.2 (2017), pp. 203–207. doi: [10.1007/s13218-016-0481-5](https://doi.org/10.1007/s13218-016-0481-5).
- [GJ94] Zoubin Ghahramani and Michael Jordan. ‘Supervised learning from incomplete data via an EM approach’. In: *Advances in Neural Information Processing Systems*. Ed. by J. Cowan, G. Tesauro, and J. Alspector. Vol. 6. Morgan-Kaufmann, 1994.
- [GK16] Lisa Gutzeit. and Elsa Andrea Kirchner. ‘Automatic Detection and Recognition of Human Movement Patterns in Manipulation Tasks’. In: *Proceedings of the 3rd International Conference on Physiological Computing Systems - PhyCS, INSTICC*. SciTePress, 2016, pp. 54–63. doi: [10.5220/0005946500540063](https://doi.org/10.5220/0005946500540063).
- [Gmb21a] DFKI GmbH. *Dual Arm Exoskeleton - Exoskeleton for upper body robotic assistance (Recupera REHA)*. 2021. url: <https://robotik.dfki-bremen.de/en/research/robot-systems/aktives-zweiarm-exos/> (visited on 09/17/2021).
- [Gmb21b] DFKI GmbH. *Mobipick Robot*. 2021. url: <https://robotik.dfki-bremen.de/en/research/robot-systems/mobipick/> (visited on 07/26/2021).
- [Gut+18] Lisa Gutzeit, Alexander Fabisch, Marc Otto, Jan Hendrik Metzen, Jonas Hansen, Frank Kirchner, and Elsa Andrea Kirchner. ‘The BesMan Learning Platform for Automated Robot Skill Learning’. In: *Frontiers in Robotics and AI* 5 (2018), p. 43. doi: [10.3389/frobt.2018.00043](https://doi.org/10.3389/frobt.2018.00043).
- [Han+18] Dong Han, Hong Nie, Jinbao Chen, and Meng Chen. ‘Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection’. In: *Robotics and Computer-Integrated Manufacturing* 49 (2018), pp. 98–104. doi: <https://doi.org/10.1016/j.rcim.2017.05.013>.
- [Har+13] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. ‘Rotation averaging’. In: *International Journal of Computer Vision* 103.3 (2013), pp. 267–305. doi: [10.1007/s11263-012-0601-0](https://doi.org/10.1007/s11263-012-0601-0).
- [HDA17] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. ‘Robot Collisions: A Survey on Detection, Isolation, and Identification’. In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1292–1312. doi: [10.1109/TR0.2017.2723903](https://doi.org/10.1109/TR0.2017.2723903).
- [HLH18] Jemin Hwangbo, Joonho Lee, and Marco Hutter. ‘Per-Contact Iteration Method for Solving Contact Dynamics’. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 895–902. doi: [10.1109/LRA.2018.2792536](https://doi.org/10.1109/LRA.2018.2792536).



- [Hop+16] Michael A. Hopkins, Alexander Leonessa, Brian Y. Lattimer, and Dennis W. Hong. 'Optimization-Based Whole-Body Control of a Series Elastic Humanoid Robot'. In: *International Journal of Humanoid Robotics* 13.1 (2016), pp. 1–34. doi: [10.1142/S0219843615500346](https://doi.org/10.1142/S0219843615500346).
- [HOt01] Nikolaus Hansen, Andreas Ostermeier, and tmp. 'Completely Derandomized Self-Adaptation in Evolution Strategies'. In: *Evolutionary Computation* 9.2 (June 2001), pp. 159–195. doi: [10.1162/106365601750190398](https://doi.org/10.1162/106365601750190398).
- [How+09] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. 'A novel method for learning policies from variable constraint data'. In: *Autonomous Robots* 27.2 (Aug. 2009), pp. 105–121. doi: [10.1007/s10514-009-9129-8](https://doi.org/10.1007/s10514-009-9129-8).
- [HRO16] Bernd Henze, Máximo A. Roa, and Christian Ott. 'Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios'. In: *The International Journal of Robotics Research* 35.12 (2016), pp. 1522–1543. doi: [10.1177/0278364916653815](https://doi.org/10.1177/0278364916653815).
- [HW96] John M. Hollerbach and Charles W. Wampler. 'The Calibration Index and Taxonomy for Robot Kinematic Calibration Methods'. In: *The International Journal of Robotics Research* 15.6 (1996), pp. 573–591. doi: [10.1177/027836499601500604](https://doi.org/10.1177/027836499601500604).
- [INS02] A.J. Ijspeert, J. Nakanishi, and S. Schaal. 'Movement imitation with nonlinear dynamical systems in humanoid robots'. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. 2002, 1398–1403 vol.2. doi: [10.1109/ROBOT.2002.1014739](https://doi.org/10.1109/ROBOT.2002.1014739).
- [Iva+12] Serena Ivaldi, Olivier Sigaud, Bastien Berret, and Francesco Nori. 'From Humans to Humanoids: the Optimal Control Framework'. In: *Paladyn Journal of Behavioral Robotics* 3(2) (June 2012), pp. 75–91. doi: [10.2478/s13230-012-0022-3](https://doi.org/10.2478/s13230-012-0022-3).
- [Jai10] Abhinandan Jain. *Robot and multibody dynamics: analysis and algorithms*. Springer Science & Business Media, 2010.
- [Kaj+03] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. 'Resolved Momentum Control: Humanoid Motion Planning Based on the Linear and Angular Momentum'. In: *IEEE International Conference on Intelligent Robots and Systems* 2 (2003), pp. 1644–1650. doi: [10.1109/iros.2003.1248880](https://doi.org/10.1109/iros.2003.1248880).
- [Kan+09] O. Kanoun, F. Lamiroux, P. B. Wieber, F. Kanehiro, E. Yoshida, and J. P. Laumond. 'Prioritizing linear equality and inequality systems: Application to local motion planning for redundant robots'. In: *2009 IEEE International Conference on Robotics and Automation*. May 2009, pp. 2939–2944. doi: [10.1109/ROBOT.2009.5152293](https://doi.org/10.1109/ROBOT.2009.5152293).
- [KB12] I. Kresse and M. Beetz. 'Movement-aware action control - Integrating symbolic and control-theoretic action execution'. In: *2012 IEEE International Conference on Robotics and Automation (ICRA)*. May 2012, pp. 3245–3251. doi: [10.1109/ICRA.2012.6225119](https://doi.org/10.1109/ICRA.2012.6225119).
- [KCC10] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell. 'Robot motor skill coordination with EM-based Reinforcement Learning'. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 3232–3237. doi: [10.1109/IROS.2010.5649089](https://doi.org/10.1109/IROS.2010.5649089).

- [Kha85] O. Khatib. ‘Real-time obstacle avoidance for manipulators and mobile robots’. In: *Proceedings 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 500–505. doi: [10.1109/R0B0T.1985.1087247](https://doi.org/10.1109/R0B0T.1985.1087247).
- [Kha87] O. Khatib. ‘A unified approach for motion and force control of robot manipulators: The operational space formulation’. In: *IEEE Journal of Robotics and Automation* 3.1 (Feb. 1987), pp. 43–53. doi: [10.1109/JRA.1987.1087068](https://doi.org/10.1109/JRA.1987.1087068).
- [Kle+21] Sébastien Kleff, Avadesh Meduri, Rohan Budhiraja, Nicolas Mansard, and Ludovic Righetti. ‘High-Frequency Nonlinear Model Predictive Control of a Manipulator’. In: *IEEE International Conference on Robotics and Automation (ICRA 2021)*. Xi’an, China, May 2021. doi: [10.1109/ICRA48506.2021.9560990](https://doi.org/10.1109/ICRA48506.2021.9560990).
- [Koo+16] Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas de Boer, Tingfan Wu, Jesper Smith, Johannes Engelsberger, and Jerry E. Pratt. ‘Design of a Momentum-Based Control Framework and Application to the Humanoid Robot Atlas’. In: *International Journal of Humanoid Robotics* 13.1 (2016). Software Documentation: [https://bitbucket.org/ihmcrobotics/ihmc\\_ros/wiki/whole-body-controller](https://bitbucket.org/ihmcrobotics/ihmc_ros/wiki/whole-body-controller), 1650007:1–1650007:34. doi: [10.1142/S0219843616500079](https://doi.org/10.1142/S0219843616500079).
- [KP10] Jens Kober and Jan Peters. ‘Imitation and Reinforcement Learning’. In: *IEEE Robotics & Automation Magazine* 17 (July 2010), pp. 55–62. doi: [10.1109/MRA.2010.936952](https://doi.org/10.1109/MRA.2010.936952).
- [KP14] Jens Kober and Jan Peters. ‘Policy Search for Motor Primitives in Robotics’. In: *Learning Motor Skills: From Algorithms to Robot Experiments*. Springer International Publishing, 2014, pp. 83–117. doi: [10.1007/978-3-319-03194-1\\_4](https://doi.org/10.1007/978-3-319-03194-1_4).
- [KPT14] Scott Kuindersma, Frank Permenter, and Russ Tedrake. ‘An efficiently solvable quadratic program for stabilizing dynamic locomotion’. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2014), pp. 2589–2594. doi: [10.1109/ICRA.2014.6907230](https://doi.org/10.1109/ICRA.2014.6907230).
- [Kui+16] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. ‘Optimization-Based Locomotion Planning, Estimation, and Control Design for the Atlas Humanoid Robot’. In: *Autonomous Robots* 40.3 (Mar. 2016), pp. 429–455. doi: [10.1007/s10514-015-9479-3](https://doi.org/10.1007/s10514-015-9479-3).
- [Kum+19a] Shivesh Kumar, Julius Martensen, Andreas Mueller, and Frank Kirchner. ‘Model Simplification For Dynamic Control of Series-Parallel Hybrid Robots - A Representative Study on the Effects of Neglected Dynamics’. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 5701–5708. doi: [10.1109/IROS40897.2019.8967786](https://doi.org/10.1109/IROS40897.2019.8967786).
- [Kum+19b] Shivesh Kumar, Abhilash Nayak, Heiner Peters, Christopher Schulz, Andreas Mueller, and Frank Kirchner. ‘Kinematic Analysis of a Novel Parallel 2SPRR+1U Ankle Mechanism in Humanoid Robot’. In: Jan. 2019, pp. 431–439. doi: [10.1007/978-3-319-93188-3\\_49](https://doi.org/10.1007/978-3-319-93188-3_49).
- [Kum+19c] Shivesh Kumar, Hendrik Wöhrle, Mathias Trampler, Marc Simnofske, Heiner Peters, Martin Mallwitz, Elsa Andrea Kirchner, and Frank Kirchner. ‘Modular Design and Decentralized Control of the Recupera Exoskeleton for Stroke Rehabilitation’. In: *Applied Sciences* 9.4 (2019). doi: [10.3390/app9040626](https://doi.org/10.3390/app9040626).



- [Kum+20] Shivesh Kumar, Kai Alexander von Szadkowski, Andreas Mueller, and Frank Kirchner. 'An Analytical and Modular Software Workbench for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots'. In: *Journal of Mechanisms and Robotics* (Jan. 2020), pp. 1–13. doi: [10.1115/1.4045941](https://doi.org/10.1115/1.4045941).
- [Kum19] Shivesh Kumar. 'Modular and Analytical Methods for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots'. PhD thesis. 2019.
- [Kup+17] Andras Kupcsik, Marc Peter Deisenroth, Jan Peters, Ai Poh Loh, Prahlad Vadakkepat, and Gerhard Neumann. 'Model-based contextual policy search for data-efficient generalization of robot skills'. In: *Artificial Intelligence* 247 (2017). Special Issue on AI and Robotics, pp. 415–439. doi: <https://doi.org/10.1016/j.artint.2014.11.005>.
- [Lei+14] Daniel Leidner, Alexander Dietrich, Florian Schmidt, Christoph Borst, and Alin Albu-Schäffer. 'Object-centered hybrid reasoning for whole-body mobile manipulation'. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 1828–1835. doi: [10.1109/ICRA.2014.6907099](https://doi.org/10.1109/ICRA.2014.6907099).
- [Lei+16] Daniel Leidner, Alexander Dietrich, Michael Beetz, and Alin Albu-Schäffer. 'Knowledge-enabled parameterization of whole-body control strategies for compliant service robots'. In: *Autonomous Robots* 40.3 (Mar. 2016), pp. 519–536. doi: [10.1007/s10514-015-9523-3](https://doi.org/10.1007/s10514-015-9523-3).
- [Lem+11] Johannes Lenburg, José de Gea Fernández, Markus Eich, Dennis Mrona, Peter Kampmann, Andreas Vogt, Achint Aggarwal, Yuping Shi, and Frank Kirchner. 'AILA - design of an autonomous mobile dual-arm robot'. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5147–5153. doi: [10.1109/ICRA.2011.5979775](https://doi.org/10.1109/ICRA.2011.5979775).
- [Lev+18] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 'Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection'. In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 421–436. doi: [10.1177/0278364917710318](https://doi.org/10.1177/0278364917710318).
- [LH09] M. de Lasa and A. Hertzmann. 'Prioritized optimization for task-space control'. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2009, pp. 5755–5762. doi: [10.1109/IR05.2009.5354341](https://doi.org/10.1109/IR05.2009.5354341).
- [LHV15] H. C. Lin, M. Howard, and S. Vijayakumar. 'Learning null space projections'. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 2613–2619. doi: [10.1109/ICRA.2015.7139551](https://doi.org/10.1109/ICRA.2015.7139551).
- [Lié77] A. Liégeois. 'Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms'. In: *IEEE Transactions on Systems, Man, and Cybernetics* 7.12 (1977), pp. 868–871. doi: [10.1109/TSMC.1977.4309644](https://doi.org/10.1109/TSMC.1977.4309644).
- [LMH10] Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. 'Feature-Based Locomotion Controllers'. In: *ACM Transactions on Graphics* 29.4 (July 2010). doi: [10.1145/1778765.1781157](https://doi.org/10.1145/1778765.1781157).
- [LP17] K.M. Lynch and F.C. Park. *Modern Robotics*. Cambridge University Press, 2017, pp. 85–89.

- [LPS14] R. Lober, V. Padois, and O. Sigaud. ‘Multiple task optimization using dynamical movement primitives for whole-body reactive control’. In: *2014 IEEE-RAS International Conference on Humanoid Robots*. Nov. 2014, pp. 193–198. doi: [10.1109/HUMANOIDS.2014.7041359](https://doi.org/10.1109/HUMANOIDS.2014.7041359).
- [LPS15] R. Lober, V. Padois, and O. Sigaud. ‘Variance modulated task prioritization in Whole-Body Control’. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015, pp. 3944–3949. doi: [10.1109/IROS.2015.7353932](https://doi.org/10.1109/IROS.2015.7353932).
- [LPS16] R. Lober, V. Padois, and O. Sigaud. ‘Efficient reinforcement learning for humanoid whole-body control’. In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. Nov. 2016, pp. 684–689. doi: [10.1109/HUMANOIDS.2016.7803348](https://doi.org/10.1109/HUMANOIDS.2016.7803348).
- [LRH17] H. C. Lin, P. Ray, and M. Howard. ‘Learning task constraints in operational space formulation’. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 309–315. doi: [10.1109/ICRA.2017.7989039](https://doi.org/10.1109/ICRA.2017.7989039).
- [LTP16] Mingxing Liu, Yang Tan, and Vincent Padois. ‘Generalized hierarchical control’. In: *Autonomous Robots* 40.1 (Jan. 2016), pp. 17–31. doi: [10.1007/s10514-015-9436-1](https://doi.org/10.1007/s10514-015-9436-1).
- [Mac+67] James MacQueen et al. ‘Some methods for classification and analysis of multivariate observations’. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.
- [Man+09] Nicolas Mansard, Olivier Stasse, Paul Evrard, and Abderrahmane Kheddar. ‘A versatile Generalized Inverted Kinematics implementation for collaborative working humanoid robots: The Stack Of Tasks’. In: *2009 International Conference on Advanced Robotics*. 2009, pp. 1–6.
- [Mas+20] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. ‘Crocodyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control’. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 2536–2542. doi: [10.1109/ICRA40945.2020.9196673](https://doi.org/10.1109/ICRA40945.2020.9196673).
- [Mat21] Encyclopedia of Mathematics. *Orthogonalization*. 2021. URL: <https://encyclopediaofmath.org/index.php?title=Orthogonalization> (visited on 12/01/2021).
- [MC07] N. Mansard and F. Chaumette. ‘Task Sequencing for High-Level Sensor-Based Control’. In: *IEEE Transactions on Robotics* 23.1 (Feb. 2007), pp. 60–72. doi: [10.1109/TR0.2006.889487](https://doi.org/10.1109/TR0.2006.889487).
- [MCR96] E. Marchand, F. Chaumette, and A. Rizzo. ‘Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing’. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS ’96*. Vol. 3. 1996, 1083–1090 vol.3. doi: [10.1109/IROS.1996.568954](https://doi.org/10.1109/IROS.1996.568954).

- [Met+08] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. 'The ICub Humanoid Robot: An Open Platform for Research in Embodied Cognition'. In: *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*. PerMIS '08. Gaithersburg, Maryland: Association for Computing Machinery, 2008, pp. 50–56. doi: [10.1145/1774674.1774683](https://doi.org/10.1145/1774674.1774683).
- [MFN06] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. 'YARP: Yet Another Robot Platform'. In: *International Journal of Advanced Robotic Systems* 3.1 (2006), p. 8. doi: [10.5772/5761](https://doi.org/10.5772/5761).
- [MK21] Dennis Mronga and Frank Kirchner. 'Learning context-adaptive task constraints for robotic manipulation'. In: *Robotics and Autonomous Systems* 141 (2021). doi: [10.1016/j.robot.2021.103779](https://doi.org/10.1016/j.robot.2021.103779).
- [MK85] Anthony A. Maciejewski and Charles A. Klein. 'Obstacle Avoidance for Kinetically Redundant Manipulators in Dynamically Varying Environments'. In: *The International Journal of Robotics Research* 4.3 (1985), pp. 109–117. doi: [10.1177/027836498500400308](https://doi.org/10.1177/027836498500400308).
- [MK88] Anthony A. Maciejewski and Charles A. Klein. 'Numerical filtering for the operation of robotic manipulators through kinematically singular configurations'. In: *Journal of Robotic Systems* 5.6 (1988), pp. 527–552. doi: <https://doi.org/10.1002/rob.4620050603>.
- [MKK09] Nicolas Mansard, Oussama Khatib, and Abderrahmane Kheddar. 'A Unified Approach to Integrate Unilateral Constraints in the Stack of Tasks'. In: *IEEE Transactions on Robotics* 25.3 (2009), pp. 670–685. doi: [10.1109/TR0.2009.2020345](https://doi.org/10.1109/TR0.2009.2020345).
- [MKP10] Katharina Mülling, Jens Kober, and Jan Peters. 'A biomimetic approach to robot table tennis'. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 1921–1926. doi: [10.1109/IR05.2010.5650305](https://doi.org/10.1109/IR05.2010.5650305).
- [Mod+16a] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi. 'Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization'. In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. Nov. 2016, pp. 101–108. doi: [10.1109/HUMANOIDS.2016.7803261](https://doi.org/10.1109/HUMANOIDS.2016.7803261).
- [Mod+16b] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi. 'Learning soft task priorities for control of redundant robots'. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 221–226. doi: [10.1109/ICRA.2016.7487137](https://doi.org/10.1109/ICRA.2016.7487137).
- [Mor+15] Federico L. Moro, Michael Gienger, Ambarish Goswami, Nikos G. Tsagarakis, and Darwin G. Caldwell. 'An attractor-based Whole-Body Motion Control (WBMC) system for humanoid robots'. In: *IEEE-RAS International Conference on Humanoid Robots* 2015-Febru. February (2015), pp. 42–49. doi: [10.1109/HUMANOIDS.2013.7029953](https://doi.org/10.1109/HUMANOIDS.2013.7029953).
- [Mro+20] Dennis Mronga, Tobias Knobloch, José de Gea Fernández, and Frank Kirchner. 'A constraint-based approach for human–robot collision avoidance'. In: *Advanced Robotics* 34.5 (2020), pp. 265–281. doi: [10.1080/01691864.2020.1721322](https://doi.org/10.1080/01691864.2020.1721322).

- [Mro21a] Dennis Mronga. *Human-Robot Coexistence Scenario*. 2021. URL: <https://robotik.dfki-bremen.de/de/mediathek/neueste-videos/mensch-roboter-koexi/> (visited on 11/25/2021).
- [Mro21b] Dennis Mronga. *Real-Time Collision Avoidance with External Objects using KCCD*. 2021. URL: <https://robotik.dfki-bremen.de/de/mediathek/neueste-videos/echtzeit-kollisionsv/> (visited on 11/25/2021).
- [Mro21c] Dennis Mronga. *Whole-Body Control of Series-Parallel Hybrid Robots*. 2021. URL: <https://cloud.dfki.de/owncloud/index.php/s/q5xAYGwPgpDsFzR> (visited on 11/25/2021).
- [Mro21d] Dennis Mronga. *Whole-Body Control on the iMRK System*. 2021. URL: <https://robotik.dfki-bremen.de/de/mediathek/neueste-videos/whole-body-control-a/> (visited on 11/22/2021).
- [MS19] Federico L. Moro and Luis Sentis. ‘Whole-Body Control of Humanoid Robots’. In: *Humanoid Robotics: A Reference*. Ed. by Ambarish Goswami and Prahlad Vadakkepat. Dordrecht: Springer Netherlands, 2019, pp. 1161–1183. DOI: [10.1007/978-94-007-6046-2\\_51](https://doi.org/10.1007/978-94-007-6046-2_51).
- [MSF22] D. Mronga, S.Kumar, and F.Kirchner. ‘Whole-Body Control of Series-Parallel Hybrid Robots’. 2022 IEEE International Conference on Robotics and Automation (ICRA), Accepted for publication. 2022.
- [ND02] Chrystopher L. Nehaniv and Kerstin Dautenhahn. ‘The Correspondence Problem’. In: *Imitation in Animals and Artifacts*. Cambridge, MA, USA: MIT Press, 2002, pp. 41–61.
- [NHY87] Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. ‘Task-priority Based Redundancy Control of Robot Manipulators’. In: *International Journal of Robotics Research* 6.2 (June 1987), pp. 3–15. DOI: [10.1177/027836498700600201](https://doi.org/10.1177/027836498700600201).
- [Nor+15] Francesco Nori, Silvio Traversaro, Jorhabib Eljaik, Francesco Romano, Andrea Del Prete, and Daniele Pucci. ‘iCub whole-body control through force regulation on rigid non-coplanar contacts’. In: *Frontiers in Robotics and AI* (Mar. 2015), p. 18. DOI: [10.3389/frobt.2015.00006](https://doi.org/10.3389/frobt.2015.00006).
- [ODA15] Christian Ott, Alexander Dietrich, and Alin Albu-Schäffer. ‘Prioritized multi-task compliance control of redundant manipulators’. In: *Automatica* 53 (2015), pp. 416–423. DOI: <http://dx.doi.org/10.1016/j.automatica.2015.01.015>.
- [OG08] David E. Orin and Ambarish Goswami. ‘Centroidal momentum matrix of a humanoid robot: Structure and properties’. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* October 2008 (2008), pp. 653–659. DOI: [10.1109/IROS.2008.4650772](https://doi.org/10.1109/IROS.2008.4650772).
- [Par+13] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. ‘Probabilistic movement primitives’. In: *Advances in Neural Information Processing Systems* (2013), pp. 1–9.
- [Pas+09] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. ‘Learning and generalization of motor skills by learning from demonstration’. In: *2009 IEEE International Conference on Robotics and Automation*. May 2009, pp. 763–768. DOI: [10.1109/ROBOT.2009.5152385](https://doi.org/10.1109/ROBOT.2009.5152385).

- [PDA19] Cristian Alejandro Vergara Perico, Joris De Schutter, and Erwin Aertbelien. ‘Combining Imitation Learning with Constraint-Based Task Specification and Control’. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1892–1899. doi: [10.1109/LRA.2019.2898035](https://doi.org/10.1109/LRA.2019.2898035).
- [Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. ‘Scikit-learn: Machine Learning in Python’. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [PKM17] Dmitry Popov, Alexandr Klimchik, and Nikolaos Mavridis. ‘Collision detection, localization and classification for industrial robots with joint torque sensors’. In: *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2017, pp. 838–843. doi: [10.1109/ROMAN.2017.8172400](https://doi.org/10.1109/ROMAN.2017.8172400).
- [PM16] A. D. Prete and N. Mansard. ‘Robustness to Joint-Torque-Tracking Errors in Task-Space Inverse Dynamics’. In: *IEEE Transactions on Robotics* 32.5 (Oct. 2016), pp. 1091–1105. doi: [10.1109/TR0.2016.2593027](https://doi.org/10.1109/TR0.2016.2593027).
- [Pol+02] N.S. Pollard, J.K. Hodgins, M.J. Riley, and C.G. Atkeson. ‘Adapting human motion for the control of a humanoid robot’. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation*. Vol. 2. 2002, 1390–1397 vol.2. doi: [10.1109/ROBOT.2002.1014737](https://doi.org/10.1109/ROBOT.2002.1014737).
- [Pro21] The Orocos Project. *Orocos Kinematics and Dynamics*. 2021. URL: <https://www.oroocos.org/kdl.html> (visited on 12/02/2021).
- [Rad+15] Nicolaus A. Radford, Philip Strawser, Kimberly Hambuchen, Joshua S. Mehling, William K. Verdeyen, A. Stuart Donnan, James Holley, Jairo Sanchez, Vienny Nguyen, Lyndon Bridgwater, Reginald Berka, Robert Ambrose, Mason Myles Markee, N. J. Fraser-Chanpong, Christopher McQuin, John D. Yamokoski, Stephen Hart, Raymond Guo, Adam Parsons, Brian Wightman, Paul Dinh, Barrett Ames, Charles Blakely, Courtney Edmondson, Brett Sommers, Rochelle Rea, Chad Tobler, Heather Bibby, Brice Howard, Lei Niu, Andrew Lee, Michael Conover, Lily Truong, Ryan Reed, David Chesney, Robert Platt Jr, Gwendolyn Johnson, Chien-Liang Fok, Nicholas Paine, Luis Sentis, Eric Cousineau, Ryan Sinnet, Jordan Lack, Matthew Powell, Benjamin Morris, Aaron Ames, and Jide Akinyode. ‘Valkyrie: NASA’s First Bipedal Humanoid Robot’. In: *Journal of Field Robotics* 32.3 (2015), pp. 397–419. doi: <https://doi.org/10.1002/rob.21560>.
- [Ras04] Carl Edward Rasmussen. ‘Gaussian Processes in Machine Learning’. In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. doi: [10.1007/978-3-540-28650-9\\_4](https://doi.org/10.1007/978-3-540-28650-9_4).
- [RCC14] Leonel Roza, Sylvain Calinon, and Darwin G. Caldwell. ‘Learning force and position constraints in human-robot cooperative transportation’. In: *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. 2014, pp. 619–624. doi: [10.1109/ROMAN.2014.6926321](https://doi.org/10.1109/ROMAN.2014.6926321).



- [RJ21] Thomas Röhr and Sylvain Joyeux. *Rock - The Robot Construction Kit*. 2021. URL: <https://www.rock-robotics.org/> (visited on 08/05/2021).
- [Rob21a] Fuzzy Logic Robotics. *ORCA Documentation*. 2021. URL: <https://orca-controller.readthedocs.io/en/master/> (visited on 07/29/2021).
- [Rob21b] Robotiq. *Robotiq 3-Finger Adaptive Robot Gripper*. 2021. URL: <https://robotiq.com/products/3-finger-adaptive-robot-gripper> (visited on 05/20/2021).
- [Saa+11] L. Saab, N. Mansard, F. Keith, J. Y. Fourquet, and P. Soueres. ‘Generation of dynamic motion for anthropomorphic systems under prioritized equality and inequality constraints’. In: *2011 IEEE International Conference on Robotics and Automation*. May 2011, pp. 1091–1096. DOI: [10.1109/ICRA.2011.5980384](https://doi.org/10.1109/ICRA.2011.5980384).
- [Saa+13] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Souères, and J. Y. Fourquet. ‘Dynamic Whole-Body Motion Generation Under Rigid Contacts and Other Unilateral Constraints’. In: *IEEE Transactions on Robotics* 29.2 (Apr. 2013), pp. 346–362. DOI: [10.1109/TR0.2012.2234351](https://doi.org/10.1109/TR0.2012.2234351).
- [SBI18] Matthew Spenko, Stephen Buerger, and Karl Iagnemma. *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*. Jan. 2018.
- [Sch+07] Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré, Ruben Smits, Erwin Aertbeliën, Kasper Claes, and Herman Bruyninckx. ‘Constraint-based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty’. In: *The International Journal of Robotics Research* 26.5 (2007), pp. 433–455. DOI: [10.1177/027836490707809107](https://doi.org/10.1177/027836490707809107).
- [Sch+18] Philipp Schmidt, Nikolaus Vahrenkamp, Mirko Wächter, and Tamim Asfour. ‘Grasping of Unknown Objects Using Deep Convolutional Neural Networks Based on Depth Images’. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6831–6838. DOI: [10.1109/ICRA.2018.8463204](https://doi.org/10.1109/ICRA.2018.8463204).
- [SGK05] Kumara Sastry, David Goldberg, and Graham Kendall. ‘Genetic Algorithms’. In: *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Ed. by Edmund K. Burke and Graham Kendall. Boston, MA: Springer US, 2005, pp. 97–125. DOI: [10.1007/0-387-28356-0\\_4](https://doi.org/10.1007/0-387-28356-0_4).
- [Sic+08] B. Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo, et al. *Robotics: Modelling, Planning and Control*. Vol. 2. Springer London, 2008.
- [Sil+19] João Silvério, Sylvain Calinon, Leonel Roza, and Darwin G. Caldwell. ‘Learning Task Priorities from Demonstrations’. In: *IEEE Transactions on Robotics* 35.1 (2019), pp. 78–94. DOI: [10.1109/TR0.2018.2878355](https://doi.org/10.1109/TR0.2018.2878355).
- [SK04] Luis Sentis and Oussama Khatib. ‘Task-oriented control of humanoid robots through prioritization’. In: *International Conference on Humanoid Robotics* (2004), pp. 1–16.
- [SK06] L. Sentis and O. Khatib. ‘A whole-body control framework for humanoids operating in human environments’. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation*. May 2006, pp. 2641–2648. DOI: [10.1109/ROBOT.2006.1642100](https://doi.org/10.1109/ROBOT.2006.1642100).
- [SLE91] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: The Task Function Approach*. Vol. 2. Oxford Engineering Science Series. Oxford: Clarendon Press, 1991.

- [Smi+08] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter. ‘iTASC: a tool for multi-sensor integration in robot manipulation’. In: *Multisensor Fusion and Integration for Intelligent Systems*, 2008. MFI 2008. *IEEE International Conference on Software Documentation*: <https://www.oroocos.org/itasc.html>. Aug. 2008, pp. 426–433. DOI: [10.1109/MFI.2008.4648032](https://doi.org/10.1109/MFI.2008.4648032).
- [Soc21] IEEE Robotics & Automation Society. *Technical Committee for Whole-Body Control*. 2021. URL: <https://www.ieee-ras.org/whole-body-control> (visited on 05/11/2021).
- [Soe+21] Peter Soetens, Takis Issaris, Herman Bruyninckx, Sylvain Joyeux, and Ruben Smits et al. *Orocos Project documentation*. 2021. URL: <https://docs.oroocos.org/> (visited on 08/05/2021).
- [Som+15] Nikhil Somani, Andre Gaschler, Markus Rickert, Alexander Perzylo, and Alois Knoll. ‘Constraint-based task programming with CAD semantics: From intuitive specification to real-time control’. In: *IEEE International Conference on Intelligent Robots and Systems* (2015), pp. 2854–2859. DOI: [10.1109/IR0S.2015.7353770](https://doi.org/10.1109/IR0S.2015.7353770).
- [Son+10] D. Song, K. Huebner, V. Kyrki, and D. Kragic. ‘Learning task constraints for robot grasping using graphical models’. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2010, pp. 1579–1585. DOI: [10.1109/IR0S.2010.5649406](https://doi.org/10.1109/IR0S.2010.5649406).
- [Spi+17] J. Spitz, K. Bouyarmane, S. Ivaldi, and J. Mouret. ‘Trial-and-error learning of repulsors for humanoid QP-based whole-body control’. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. Nov. 2017, pp. 468–475. DOI: [10.1109/HUMAN0IDS.2017.8246914](https://doi.org/10.1109/HUMAN0IDS.2017.8246914).
- [SS88] Lorenzo Sciavicco and Bruno Siciliano. ‘A Solution Algorithm to the Inverse Kinematic Problem for Redundant Manipulators’. In: *IEEE Journal on Robotics and Automation* 4.4 (1988), pp. 403–410. DOI: [10.1109/56.804](https://doi.org/10.1109/56.804).
- [SS91] B. Siciliano and J.-J.E. Slotine. ‘A general framework for managing multiple tasks in highly redundant robotic systems’. In: *Fifth International Conference on Advanced Robotics ‘Robots in Unstructured Environments*. 1991, 1211–1216 vol.2. DOI: [10.1109/ICAR.1991.240390](https://doi.org/10.1109/ICAR.1991.240390).
- [Sys21] Robot Operating System. *URDF - Unified Robot Description Format*. 2021. URL: <http://wiki.ros.org/urdf> (visited on 05/18/2021).
- [TBB14] Moritz Tenorth, Georg Bartels, and Michael Beetz. ‘Knowledge-Based Specification of Robot Motions’. In: *Proceedings of the Twenty-First European Conference on Artificial Intelligence*. ECAI’14. Prague, Czech Republic: IOS Press, 2014, pp. 873–878.
- [TBF11] Holger Täubig, Berthold Bäuml, and Udo Frese. ‘Real-time swept volume and distance computation for self collision detection’. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 1585–1592. DOI: [10.1109/IR0S.2011.6094611](https://doi.org/10.1109/IR0S.2011.6094611).
- [TD19] Russ Tedrake and the Drake Development Team. *Drake: Model-based design and verification for robotics*. 2019. URL: <https://drake.mit.edu>.



- [TET12a] Yuval Tassa, Tom Erez, and Emanuel Todorov. ‘Synthesis and stabilization of complex behaviors through online trajectory optimization’. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 4906–4913. doi: [10.1109/IR0S.2012.6386025](https://doi.org/10.1109/IR0S.2012.6386025).
- [TET12b] E. Todorov, T. Erez, and Y. Tassa. ‘MuJoCo: A physics engine for model-based control’. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2012, pp. 5026–5033. doi: [10.1109/IR0S.2012.6386109](https://doi.org/10.1109/IR0S.2012.6386109).
- [TF12] H. Taeubig and U. Frese. ‘A New Library for Real-time Continuous Collision Detection’. In: *Proceedings of ROBOTIK 2012; 7th German Conference on Robotics*. May 2012, pp. 1–5.
- [TL96] S.K. Tso and K.P. Liu. ‘Hidden Markov model for intelligent extraction of robot trajectory command from demonstrated trajectories’. In: *Proceedings of the IEEE International Conference on Industrial Technology (ICIT’96)*. 1996, pp. 294–298. doi: [10.1109/ICIT.1996.601593](https://doi.org/10.1109/ICIT.1996.601593).
- [Tod06] Emanuel Todorov. ‘Optimal Control Theory’. In: *Bayesian Brain: Probabilistic Approaches to Neural Coding*. Dec. 2006. doi: [10.7551/mitpress/9780262042383.003.0012](https://doi.org/10.7551/mitpress/9780262042383.003.0012).
- [Tou21] Marc Toussaint. *Lecture Notes: Gaussian identities*. 2021. URL: <https://ipvs.informatik.uni-stuttgart.de/mlr/marc/notes/gaussians.pdf> (visited on 11/23/2021).
- [Tur98] Roy M. Turner. ‘Context-mediated behavior for intelligent agents’. In: *International Journal of Human Computer Studies* 48.3 (1998), pp. 307–330. doi: [10.1006/ijhc.1997.0173](https://doi.org/10.1006/ijhc.1997.0173).
- [Ude+10] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. ‘Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives’. In: *IEEE Transactions on Robotics* 26.5 (2010), pp. 800–815. doi: [10.1109/TR0.2010.2065430](https://doi.org/10.1109/TR0.2010.2065430).
- [Ude+14] Aleš Ude, Bojan Nemec, Tadej Petrić, and Jun Morimoto. ‘Orientation in Cartesian space dynamic movement primitives’. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2997–3004. doi: [10.1109/ICRA.2014.6907291](https://doi.org/10.1109/ICRA.2014.6907291).
- [Uni21] Human-Centered Robotics Lab of the University of Texas at Austin. *ControlIt! - A Whole Body Operational Space Control Middleware*. 2021. URL: <https://github.com/liangfok/controlit> (visited on 07/29/2021).
- [Ure+15] Ana Lucia Pais Ureche, Keisuke Umezawa, Yoshihiko Nakamura, and Aude Billard. ‘Task Parameterization Using Continuous Constraints Extracted from Human Demonstrations’. In: *IEEE Transactions on Robotics* 31.6 (2015), pp. 1458–1471. doi: [10.1109/TR0.2015.2495003](https://doi.org/10.1109/TR0.2015.2495003).
- [Vai+15] Joris Vaillant, Abderrahmane Kheddar, Herve Audren, Francois Keith, Stanislas Brossette, Kenji Kaneko, Mitsuharu Morisawa, Eiichi Yoshida, and Fumio Kanehiro. ‘Vertical ladder climbing by the HRP-2 humanoid robot’. In: *IEEE-RAS International Conference on Humanoid Robots* 2015-February. November (2015), pp. 671–676. doi: [10.1109/HUMAN0IDS.2014.7041435](https://doi.org/10.1109/HUMAN0IDS.2014.7041435).

- [Van+12] Dominick Vanthienen, Tinne De Laet, Herman Bruyninckx, Wilm Decré, and Joris De Schutter. 'Force-Sensorless and Bimanual Human-Robot Comanipulation Implementation using iTaSC'. In: *IFAC Proceedings Volumes* 45.22 (2012), pp. 759–766. doi: <https://doi.org/10.3182/20120905-3-HR-2030.00127>.
- [VDS05] Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. 'Incremental Online Learning in High Dimensions'. In: *Neural Computation* 17.12 (Dec. 2005), pp. 2602–2634. doi: [10.1162/089976605774320557](https://doi.org/10.1162/089976605774320557).
- [Whi69] Daniel E. Whitney. 'Resolved Motion Rate Control of Manipulators and Human Prostheses'. In: *IEEE Transactions on Man-Machine Systems* 10.2 (1969), pp. 47–53. doi: [10.1109/TMMS.1969.299896](https://doi.org/10.1109/TMMS.1969.299896).
- [WLP17] D. Wilbers, R. Lioutikov, and J. Peters. 'Context-driven movement primitive adaptation'. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 3469–3475. doi: [10.1109/ICRA.2017.7989396](https://doi.org/10.1109/ICRA.2017.7989396).
- [Xin15] Ben Xinjilefu. 'State Estimation for Humanoid Robots'. PhD thesis. Pittsburgh, PA: Carnegie Mellon University, Aug. 2015.
- [Yos85] T. Yoshikawa. 'Manipulability and redundancy control of robotic mechanisms'. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 1004–1009. doi: [10.1109/ROBOT.1985.1087283](https://doi.org/10.1109/ROBOT.1985.1087283).