

**Towards Autonomous Policy Synthesis:  
Tactile Manipulation Skills, Learning  
Architecture, and  
Process-Taxonomy-Based Planning**

Lars Johannsmeier, M.Sc.

**Ph.D. Thesis**



**TECHNISCHE UNIVERSITÄT MÜNCHEN**  
**TUM School of Computation, Information and Technology**

**Towards Autonomous Policy Synthesis: Tactile  
Manipulation Skills, Learning Architecture, and  
Process-Taxonomy-Based Planning**

Lars Johannsmeier, M.Sc.

Vollständiger Abdruck der von der TUM School of Computation, Information and  
Technology der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Angela Schoellig

Prüfende der Dissertation: 1. Prof. Dr.-Ing. Sami Haddadin

2. Prof. Dr. Marc Toussaint

3. Prof. Dr. Dieter Fox

Die Dissertation wurde am **18.04.2023** bei der Technischen Universität München ein-  
gereicht und durch die TUM School of Computation, Information and Technology am  
**22.12.2023** angenommen.



Will robots inherit the earth? Yes,  
but they will be our children.

---

*Marvin Minsky*



# Acknowledgment

Human intelligence is most noteworthy in our ability to design and use tools. It is my dream to make robots become intelligent tools, the hammer of tomorrow, an extension of our own mind. This thesis is my humble attempt to make this dream come true, and thus contribute to solutions for our society's most pressing problems by utilizing robots as the physical embodiment of artificial intelligence.

For the opportunity to complete this work, first and foremost I want to thank my supervisor and mentor Sami Haddadin for his continued support, counsel and feedback during the process of creating this dissertation, as well as for the numerous opportunities to explore various facets of the robotics research landscape. He gave me the opportunity to do research on the very frontier of robotics technology, and the platform to present my work to the research community, industry, politics and beyond. The impact that my work has might otherwise not have been possible.

Then, I would like to thank my former colleagues at the Munich Institute of Robotics and Machine Intelligence as well as the Institute of Automatic Control at the Gottfried Wilhelm Leibniz Universität Hannover, Anna Adamczyk, Lingyun Chen, Xiao Chen, Diego Hidalgo Carvajal, Alexander Moortgat-Pick, Robin Kirschner, Torsten Lilge, Kim Peper, Edmundo Pozo Fortunic, Dennis Ossadnik, Johannes Ringwald, Moritz Schappler, Jonathan Vorndamme, and Fan Wu for their company, interesting conversations and perspectives, and fruitful collaboration. I especially want to express my gratitude to Samuel Schneider who supported me with experimental work, Fernando Diaz Ledezma, Dennis Knobbe, Johannes Kühn, Erfan Shahriari, Florian Voigt, and Peter So for the many successful team-ups for publications and demonstrators, Rolando Franjga for the invaluable craftsmanship support and pragmatism, and Jan Harder and Regine Hunstein for their commitment and dedication to make MIRMI a place of opportunities that enabled me to write this thesis. Furthermore, I want to thank Sven Parusel at Franka Emika GmbH for his continued support with the robot hardware I used.

Finally, I would like to thank my parents, my brother and my sister for their support and interest in my work, and most importantly my wife Maria and daughter Kira for their love, support, and patience.





# Abstract

Robots are becoming more and more commodity in our society. Industry has long been a place for robots, but now also healthcare, logistics, and the domestic sector are starting to take advantage of such flexible automation solutions. The most relevant drivers for this development are the demographic change and labor shortage. In order to meet the upcoming requirements technology jumps are necessary that pave the way for next-generation tactile robots. Future Workplaces and factories will rely on the versatility and manipulation capabilities of tactile robots to automate highly dynamic manufacturing processes while keeping the energy demand low. The robots will generate solutions to problems on-the-fly without the need for manual programming, and learn them completely autonomously. They will become intelligent tools in autonomous, modular factories, assistants in households, and embodiments for telepresence applications. Besides interaction and mobility, manipulation capabilities are key to progress further in this direction.

Vast progress in robotic manipulation research has already been made toward this vision, especially in the last decade, and the results, such as novel control methodologies, motion planning, and perception are beginning to merge into the industrial domain. As of today there already exist application fields such as testing, inspection and machine tending that make use of advanced pre-programmed robot skills. However, considering more difficult manipulation processes, there is a two-part challenge. First, there is a missing constructive link between process descriptions and tactile skill models. Second, learning such models for challenging tactile processes is difficult and currently not even the most powerful (traditional) end-to-end frameworks are capable of generating reliable manipulation skill solutions, let alone achieve transfer between different problems.

The main contribution of this thesis is an autonomous synthesis framework that connects formal process definitions provided by process experts with compatible tactile skill models. The connection between these endpoints is proposed to be a tactile manipulation skill taxonomy that determines a unique solution skill for a given process. The tactile skill is then formulated as a learning problem that can be solved with a sample-efficient learning architecture. This divide and conquer approach can be scaled indefinitely in terms of numbers of skills and processes and can also be connected to automatic process planning systems. It may thus allow for solving complex tasks without the need for expert programming. The resulting versatility in terms of manipulation process solutions makes this concept a first step toward an ever increasing curriculum for robots that, similar to established curricula for (human) trainees in industrial fields, could provide a framework for the acquisition of all relevant manipulation skills for robots.

This thesis provides the theoretical foundations for tactile skills, their synthesis, learning and planning capabilities. It describes the implementation of the foundations and a number of validation cases. Extensive experimental work is presented that demonstrates the manipulation framework's capabilities. Exhaustive verification experiments validate the

approach for a meaningful number of skills, showcasing high robustness and performance, which are fundamental requirements for industrial applications. The learning performance of the proposed architecture and synthesis pipeline is analyzed for state-of-the-art machine learning algorithms. The overall system shows superior learning performance and efficiency when compared against bleeding-edge end-to-end approaches. In a large experimental campaign, even a systematic transfer learning effect between different challenging physical manipulation skill instances was observed. This allows to learn large number of skills by systematically exploiting their similarity. The performance of the optimized skills and the learning performance are directly compared to human capabilities. The results show that for some skills human-level performance can already be achieved. Finally, a collaborative assembly planning problem for industrial mechatronics systems with tight tolerances and multi-dimensional insertion processes showcases the planning capabilities of the framework. With the developed framework it finally is possible to automate robot programming even for complex manipulation processes without the need for robot expert knowledge.

The results of this work have impacted further research efforts in control, learning, telepresence, and motion planning, and have even opened up an entirely new field, namely dentronics. Furthermore, it has influenced various projects, publications and products. To the best of the author's knowledge, this thesis is the first that enables non-experts to solve complex manipulation problems on an industrial level through an automatic skill synthesis and learning approach with low energy demand and only restricted computational resources.

# Zusammenfassung

Roboter werden in unserer Gesellschaft immer mehr zu einem alltäglichen Werkzeug. In der Industrie werden Roboter schon lange eingesetzt, aber auch im Gesundheitswesen, in der Logistik und im häuslichen Bereich werden solche flexiblen Automatisierungslösungen zunehmend genutzt. Die wichtigsten Treiber für diese Entwicklung sind der demografische Wandel und der Arbeitskräftemangel. Um den kommenden Anforderungen gerecht zu werden, sind Technologiesprünge notwendig, die den Weg für die nächste Generation taktiler Roboter ebnen. Zukünftige Arbeitsplätze und Fabriken werden sich auf die Vielseitigkeit und die Manipulationsfähigkeiten von taktilen Robotern verlassen, um hochdynamische Fertigungsprozesse zu automatisieren und gleichzeitig den Energiebedarf niedrig zu halten. Die Roboter werden im laufenden Betrieb Lösungen für Probleme generieren, ohne dass eine manuelle Programmierung erforderlich ist, und diese völlig autonom erlernen. Sie werden zu intelligenten Werkzeugen in autonomen, modularen Fabriken, zu Assistenten in Haushalten und zu Verkörperungen für Telepräsenz Anwendungen. Neben Interaktion und Mobilität sind die Manipulationsfähigkeiten der Schlüssel zu weiteren Fortschritten in dieser Richtung.

Auf dem Gebiet der Roboter Manipulation wurden, vor allem in den letzten zehn Jahren, bereits große Fortschritte erzielt, und die Ergebnisse, wie z. B. neuartige Kontrollmethoden, Bewegungsplanung und Wahrnehmung, finden allmählich Eingang in den industriellen Bereich. Heute gibt es bereits Anwendungsbereiche wie Testen, Inspektion und Maschinenbedienung, in denen fortgeschrittene, jedoch vorprogrammierte Roboterfähigkeiten zum Einsatz kommen. Betrachtet man jedoch schwierigere Manipulationsprozesse, so besteht eine zweiteilige Herausforderung. Erstens fehlt eine konstruktive Verbindung zwischen Prozessbeschreibungen und taktilen Fähigkeitsmodellen. Zweitens ist das Erlernen solcher Modelle für anspruchsvolle taktile Prozesse schwierig, und derzeit sind nicht einmal die leistungsstärksten (traditionellen) End-to-End-Frameworks in der Lage, zuverlässige Lösungen für Manipulationsfähigkeiten zu generieren, geschweige denn eine Übertragung zwischen verschiedenen Problemen zu erreichen.

Der Hauptbeitrag dieser Arbeit ist eine autonomes Synthese-Pipeline, welche formale Prozessdefinitionen, die von Prozessexperten bereitgestellt werden, mit kompatiblen taktilen Fähigkeitsmodellen verbindet. Als Verbindung zwischen diesen Endpunkten wird eine Taxonomie der taktilen Manipulationsfähigkeiten vorgeschlagen, die eine eindeutige Lösungsfähigkeit für einen bestimmten Prozess bestimmt. Die taktile Fertigkeit wird dann als ein Lernproblem formuliert, das mit einer effizienten Lernarchitektur gelöst werden kann. Dieser teile-und-herrsche-Ansatz lässt sich in Bezug auf die Anzahl der Fertigkeiten und Prozesse unbegrenzt skalieren und kann auch mit automatischen Prozessplanungssystemen verbunden werden. Auf diese Weise können komplexe Aufgaben gelöst werden, ohne dass eine Programmierung durch Experten erforderlich ist. Die sich daraus ergebende Vielseitigkeit bei der Lösung von Manipulationsprozessen macht dieses

Konzept zu einem ersten Schritt in Richtung eines immer umfangreicheren Curriculums für Roboter, das, ähnlich wie etablierte Curricula für (menschliche) Auszubildende in industriellen Bereichen, einen Rahmen für den Erwerb aller relevanten Manipulationsfähigkeiten für Roboter bieten könnte.

Diese Arbeit liefert die theoretischen Grundlagen für taktile Fähigkeiten, ihre Synthese, Lern- und Planungsfähigkeiten. Sie beschreibt die Umsetzung der Grundlagen und eine Reihe von Validierungsfällen. Es werden umfangreiche experimentelle Arbeiten vorgestellt, die die Fähigkeiten des Manipulationsrahmens demonstrieren. Ausführliche Verifizierungsexperimente validieren den Ansatz für eine aussagekräftige Anzahl von Fertigkeiten und zeigen eine hohe Robustheit und Leistung, die grundlegende Anforderungen für industrielle Anwendungen sind. Die Lernleistung der vorgeschlagenen Architektur und Synthesepipeline wird für modernste maschinelle Lernalgorithmen analysiert. Das Gesamtsystem zeigt eine überlegene Lernleistung und Effizienz im Vergleich zu den modernsten End-to-End-Ansätzen. In einer umfangreichen experimentellen Kampagne wurde sogar ein systematischer Transfer-Lerneffekt zwischen verschiedenen anspruchsvollen physikalischen Manipulationsfähigkeiten beobachtet. Dies ermöglicht das Erlernen einer großen Anzahl von Fertigkeiten durch systematische Ausnutzung ihrer Ähnlichkeit. Die Leistung der optimierten Fertigkeiten und die Lernleistung werden direkt mit den menschlichen Fähigkeiten verglichen. Die Ergebnisse zeigen, dass für einige Fertigkeiten bereits das Niveau menschlicher Fähigkeiten erreicht werden kann. Abschließend wird anhand eines kollaborativen Montageplanungsproblems für industrielle Mechatroniksysteme mit engen Toleranzen und mehrdimensionalen Einfügeprozessen die Planungsfähigkeit des Frameworks demonstriert. Mit dem entwickelten Framework ist es schließlich möglich, die Roboterprogrammierung auch für komplexe Manipulationsprozesse zu automatisieren, ohne dass Roboterexpertenwissen erforderlich ist.

Die Ergebnisse dieser Arbeit haben weitere Forschungsarbeiten im Bereich der Steuerung, des Lernens, der Telepräsenz und der Bewegungsplanung beeinflusst und sogar ein völlig neues Gebiet, nämlich die Dentronik, eröffnet. Darüber hinaus haben sie verschiedene Projekte, Veröffentlichungen und Produkte beeinflusst. Nach bestem Wissen des Autors ist diese Arbeit die erste, die es Nicht-Experten ermöglicht, komplexe Manipulationsprobleme auf industrieller Ebene durch einen automatischen Synthese- und Lernansatz mit geringem Energiebedarf und nur begrenzten Rechenressourcen zu lösen.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Problem Statement . . . . .	1
1.3	Research Questions and Contribution . . . . .	3
1.4	State of the Art . . . . .	5
1.4.1	Robot Platforms . . . . .	5
1.4.2	Interaction Control . . . . .	5
1.4.3	Skill Frameworks . . . . .	9
1.4.4	Skill Taxonomies . . . . .	13
1.4.5	Skill Learning . . . . .	14
1.4.6	Robot Software Frameworks . . . . .	19
1.4.7	Assembly Planning . . . . .	19
1.5	Thesis Structure . . . . .	20
1.6	Curriculum Vitae . . . . .	20
<b>2</b>	<b>Theoretical Foundations</b>	<b>25</b>
2.1	Representation . . . . .	25
2.1.1	Tactile Skill . . . . .	25
2.1.2	Process Definition . . . . .	29
2.1.3	Taxonomy Structure . . . . .	30
2.2	Tactile Skill Synthesis Procedure . . . . .	31
2.3	Artificial Intelligence-Based Assembly Planning . . . . .	33
2.3.1	Introduction . . . . .	33
2.3.2	Assembly Plan Representation . . . . .	36
2.3.3	Task Allocation and Planning . . . . .	37
2.4	Machine Learning . . . . .	43
2.4.1	Algorithms . . . . .	43
2.4.2	Performance Metrics . . . . .	47
2.4.3	Robot Motor Memory Effect . . . . .	50
2.5	Conclusion . . . . .	51
<b>3</b>	<b>System Architecture and Validation Cases</b>	<b>53</b>
3.1	GGTWreP Framework . . . . .	53
3.1.1	Implementation Examples . . . . .	56
3.2	Machine Intelligence Operation System . . . . .	59
3.2.1	System Overview . . . . .	59
3.2.2	Design Objectives . . . . .	60

3.2.3	Capabilities . . . . .	61
3.2.4	Modules . . . . .	61
3.2.5	GGTWreP Implementation . . . . .	62
3.2.6	Learning . . . . .	66
3.3	Validation Experiments: Use Case Integration . . . . .	67
3.3.1	New Application Domain: Dentronics . . . . .	67
3.3.2	Distributed Control: Telepresence . . . . .	68
3.3.3	Local Multi-Robot System: Pinakothek der Moderne . . . . .	69
3.3.4	Collaborative Assembly Station . . . . .	70
3.4	Validation Experiments: Learning . . . . .	71
3.4.1	Manipulation Learning: Peg-in-Hole . . . . .	71
3.4.2	Distributed Multi-Robot System: Collective . . . . .	72
3.5	Conclusion . . . . .	73
<b>4</b>	<b>Experimental Analysis</b>	<b>75</b>
4.1	Taxonomy Verification . . . . .	76
4.1.1	Experimental Setup . . . . .	76
4.1.2	Verification Process . . . . .	76
4.1.3	Results . . . . .	77
4.2	Tactile Skill Learning . . . . .	80
4.2.1	Comparative Analysis of Algorithms for Skill Learning . . . . .	80
4.2.2	Comparison with Deep Reinforcement Learning . . . . .	83
4.2.3	Skill Transfer Learning . . . . .	86
4.3	Performance Comparison: Robot vs. Human . . . . .	93
4.3.1	Task Description . . . . .	93
4.3.2	Case Study . . . . .	98
4.4	Collaborative Assembly Planning . . . . .	101
4.4.1	Experimental Setup . . . . .	101
4.4.2	Results . . . . .	103
4.5	Conclusion . . . . .	104
<b>5</b>	<b>Conclusion</b>	<b>105</b>
5.1	Contributions . . . . .	105
5.2	Impact . . . . .	107
5.3	Future Work . . . . .	108
	<b>Appendix</b>	<b>109</b>
<b>A</b>	<b>Appendix</b>	<b>111</b>
A.1	Skill Synthesis: Policies . . . . .	111
	<b>Bibliography</b>	<b>125</b>

# Abbreviations and Symbols

In this thesis, scalar quantities are written as plain letters, e.g.,  $\lambda$ ,  $c$ ,  $K$ . Vectors and matrices are represented by bold letters, whereas vectors have small letters, e.g.,  $\mathbf{q}$ ,  $\boldsymbol{\theta}$  and matrices capital letters  $\mathbf{K}$ ,  $\mathbf{M}$ . The total derivative w.r.t. time is indicated by a dot above the symbol, i.e.,  $\dot{\mathbf{x}} = \frac{d}{dt}\mathbf{x}$ ,  $\ddot{\mathbf{x}} = \frac{d^2}{dt^2}\mathbf{x}$ . The Euclidian norm of a vector  $q$  is denoted by  $|q|$ , the dot product of two vectors  $a$  and  $b$  by  $(a, b)$  and their cross product by  $a \times b$ . All symbols are introduced in the text before they are used. In some sections, the argument is omitted for brevity. Several variables appear with different subscripts, superscripts, additional symbols and dimensions. In the following list, the quantities are generally described without being further specified. The specific meaning becomes apparent when the respective variable is introduced in the text. Please note that the list of symbols is not complete, but it contains symbols that appear frequently or are of major importance in this thesis.

## List of Symbols

### General Symbols

$\mathbf{q}$	Joint positions
$\dot{\mathbf{q}}$	Joint velocities
$\ddot{\mathbf{q}}$	Joint acceleration
$\mathbf{T}$	Pose in matrix form
$\mathbf{T}_d$	Desired pose in matrix form
$\mathbf{x}$	Pose in vector form
$\mathbf{x}_d$	Desired pose in vector form
$\dot{\mathbf{x}}$	Twist
$\dot{\mathbf{x}}_d$	Desired twist
$\ddot{\mathbf{x}}$	Cartesian acceleration
$\ddot{\mathbf{x}}_d$	Desired Cartesian acceleration
$\mathbf{f}$	Wrench
$\mathbf{f}_d$	Desired wrench
$\mathbf{f}_{ff}$	Feed-forward wrench
$\mathbf{f}_{\text{ext}}$	External wrench

$\boldsymbol{\tau}$	Joint motor torques
$\boldsymbol{\tau}_d$	Desired joint motor torques
$\boldsymbol{\tau}_{\text{ext}}$	External joint torque
$\mathbf{K}_x$	Cartesian stiffness
$\mathbf{K}$	Joint stiffness
$\mathbf{D}_x$	Cartesian damping
$\mathbf{D}$	Joint damping
$\mathbf{J}_x$	Jacobian
$\mathbf{M}$	Mass matrix
$\mathbf{C}$	Coriolis vector
$\mathbf{g}$	Gravity vector
$t$	Time
$\mathcal{U}(\cdot)$	Environment around a pose

## Process, Tactile Skill, and Taxonomy

$\varsigma$	Skill
$\iota$	Task
$\boldsymbol{\pi}_d$	Tactile policy
$\Pi$	Set of tactile policies
$p$	Process
$\mathcal{P}$	Set of processes
$\mathcal{T}$	Taxonomic synthesis algorithm
$o$	Object
$\mathcal{O}$	Set of objects
$s$	Process state
$s_0$	Initial process state
$s_1$	Final process state
$s_e$	Error process state
$s_\pi$	Policy process state
$\mathcal{C}_{\text{pre}}$	Process pre condition
$\mathcal{C}_{\text{err}}$	Process error condition
$\mathcal{C}_{\text{suc}}$	Process success condition
$\delta$	Process step transition



$\Delta$	Set of process step transition
$\theta$	Parameters
$\theta_\pi$	Skill parameters
$\theta_c$	Controller parameters

## GGTWreP framework

$w$	World state
$\mathcal{W}$	World state space
$E$	Effects of skill execution on environment
$\mathcal{R}$	Requirements for skill execution
$\mathbb{D}$	Parameter domain
$\mathbb{C}$	Skill constraints
$\mathfrak{D}$	Dataset from a learning experiment
$\mathcal{Q}$	Quality metric
$\mathcal{C}$	Tactile policy commands

## Planning

$\mathcal{A}$	Assembly
$\rho$	Assembly part
$\Gamma$	Set of assembly parts
$a$	Assembly action
$\alpha$	Sequence of assembly actions
$w$	Agent
$W$	Set of agents
$\zeta$	Assembly state
$\mathcal{Z}$	Set of assembly states

## Abbreviations

AI	Artificial intelligence
API	Application programming interface
DL	Deep learning

DLR	Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Centre)
DMP	Dynamic motion primitives
DOF	Degrees of freedom
EE	End effector
e.g.	<i>Exemplis grata</i> (for example)
ET	Empirical transferability
etc.	<i>Et cetera</i> (and so forth)
F/T sensor	Force/torque sensor
FCI	Franka control interface
GGTWreP	Graph-Guided Twist-Wrench Policy Approximation
ICRA	International Conference on Robotics and Automation
i.e.	<i>Id est</i> (that is)
IEEE	Institute of Electrical and Electronics Engineers
IROS	International Conference on Intelligent Robots and Systems
LbD	Learning by demonstration
LE	Learning effort
LER	Learning effort ratio
LTE	Long-term evolution
LWR	Lightweight Robot
KPI	Key performance indicator
ML	Machine learning
MIRMI	Munich Institute of Robotics and Machine Intelligence
ODE	Ordinary differential equation
PA	Policy approximator
PC	Personal computer
PbD	Programming by demonstration
PDDL	Planning domain definition language
RL	Reinforcement learning
RMM	Robot motor memory
ROI	Region of interest
ROS	Robot Operating System
RPC	Remote procedure call
SVM	Support vector machine
TMS	Taxonomy of Manipulation Skills

TUM	Technical University Munich
UAV	Unmanned aerial vehicle
UDP	User datagram protocol
w.r.t.	With respect to



# 1

## Introduction

### 1.1 Motivation

Currently, companies from various industrial domains are evolving from using monolithic process structures and production technologies to being dynamic and flexible service providers. The ultimate goal after the wave of industry 4.0 of the previous decade based on the major breakthroughs in data-driven artificial intelligence (AI) and robotics has become the concept of Production-as-a-service. The basic concept is that a blueprint of a product to be manufactured can be sent to a smart, modular factory system that reconfigures its production and logistics lines to manufacture the potentially low-volume product. The capability to produce virtually anything in one place requires highly flexible, autonomous tools, e.g. tactile robots with the ability to learn and generalize in short cycles, that can safely and purposefully interact with the environment and manipulate it. It also means that a continuous stream of new manufacturing processes has to be solved by these robots to cope with countless new assembly problems, custom material processing and testing requirements. Clearly, it is impossible to meet this challenge with human experts that program the robots with new skills. Instead the robots need to be able to instantiate and optimize new skills on their own from simplified instructions or observing humans.

### 1.2 Problem Statement

In robotics research, remarkable progress was achieved in recent years with milestones such as the emergence of lightweight robots [1–5], compliant interaction control [6–9] and the establishment of safe physical interaction between robots and humans [10, 11]. These developments paved the way for robotic manipulation, which is a complex challenge connecting numerous research fields such as robot hardware development, motion and interaction control, interaction policy design, vision, motion and task planning, learning and human-robot interaction. The advances in these fields brought about systematic

approaches for implementing skillful and versatile physical robot manipulation capabilities called manipulation skills. When considering near-future automated workplaces and the significant performance steps made [12, 13], manipulation skills have gained increasing interest from various application domains. In industrial workshops workers will use intuitive-to-handle robots to do repetitive work [14], in doctors' offices robot assistants will improve hygiene and reliability of diagnostics [15, 16], and the homes of elderly citizens will be equipped with robot assistants to help manage their daily routine [17, 18]. All these robotic applications require sophisticated tactile capabilities to enable a safe, reliable and efficient integration into human-centered processes. However, while workers can rely on systematic curricula specifically developed for their profession (e.g. in Germany [19]), there is no such curriculum for robots to learn the capabilities required for their tasks. A significant challenge for today's researchers and engineers is to transfer skill frameworks to relevant industrial scenarios in order to enable the automation of many tasks that are still manually performed. There are strict requirements for robot manipulation coming from this context such as safety regulations, a high degree of reliability and robustness, repeatability, reaction to unforeseen events, etc. These requirements can, at least in principle, be addressed by structured skill frameworks. However, a typical caveat of structured approaches is the missing compatibility with (machine) learning methods. As a consequence, skill implementations usually require extensive manual design and robotic expertise. The currently popular end-to-end approaches to skill modeling [20, 21] focus on this specific problem, when autonomous learning is essentially hard-wired into their architecture. However, considerable obstacles for the integration of autonomously learned robot skills into industrial processes include the need to collect massive amounts of data, and that these methods may not be able to reliably meet process constraints (in particular safety regulations) without significant intervention in their internal structure. Furthermore, the required learning time, achievable performance, and amount of computational and energy resources needed are still far from being practical [22]. These limitations must be overcome before large numbers of robots will be able to perform countless manipulation skills in different situations. In particular, it would be impractical for each robot to learn each skill from scratch. It is also impractical for human experts to manually program each robot, let alone thousands of devices, which is still today's preferred solution in industrial automation. The ability to efficiently transfer knowledge from already learned skills to solve new problems would significantly advance robot learning and improve the manipulation performance of robots, allowing them to complete tasks to the high standards that are required in real-world scenarios. Being able to use prior experience to significantly speed up this learning process would make robots not only more resource-efficient, but also much more flexible to use. In the end, current approaches are not versatile enough to match the need for highly flexible automation solutions in today's industry. Structure-based frameworks are often able to implement many different skills but there is so far no way to systematically scale their designs to large numbers of processes. Similarly, learning-based frameworks lack proper transfer capabilities such that extensive learning for each skill is still required. An additional problem external to the skill frameworks themselves is the necessity to identify appropriate tactile manipulation skills for each specific physical process (including goals, constraints, subtasks, etc.) without access to robot expert knowledge. This issue prevents laypeople, e.g. shop-floor workers or technicians, from employing robots to carry out repetitive or demanding tasks, as there generally is a lack of expertise in the implementation of robot manipulation skills. As a result, compa-

nies either rely on manual labor, or have to find and pay rare and expensive experts to set up automation solutions. As mentioned above, autonomous learning capabilities, at least when used alone, are not yet ready to fill this gap.

### 1.3 Research Questions and Contribution

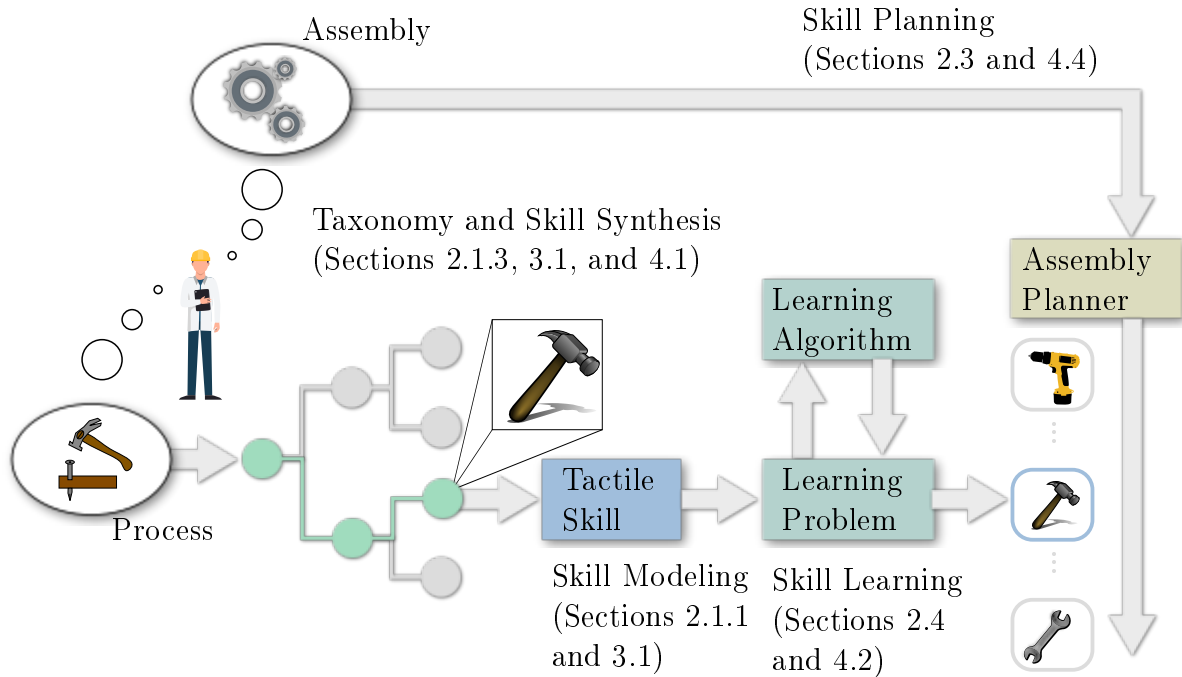
The set of fundamental research questions that are addressed in this thesis aim to provide convincing answers to help push the existing boundaries of robotics research and generate solutions to several of the aforementioned challenges:

- Q1: How can robot expert knowledge be systematically encoded such that non-experts can use robots to automate complex manipulation processes?
- Q2: For this, it is essential to understand how tactile manipulation skills can be encoded such that control, policy, learning, and planning are unified.
- Q3: To generate breakthroughs in the robot learning problem, it needs to be understood how complex robot manipulation skills can be learned in a short amount of time and with low energy consumption and computational demands such that the solution is robust and of high performance.
- Q4: Finally, this thesis aims to give answers to the question of whether systematic similarities between skills can be exploited by new ways of transfer learning so skill learning can be scaled up to large numbers of skills without running into the curse of dimensionality.

This thesis has made three core contributions that provide answers to above research questions. The contributions can be clustered into three major groups, namely tactile skill modeling, automatic skill synthesis, and autonomous skill learning. In Fig. 1.1 they are summarized in an overview.

**Contribution 1: Tactile Skill Modeling** The theoretical foundations for tactile skills are provided, consisting of tactile platform, tactile controller, tactile policy, and performance evaluator. The graph-guided twist-wrench policy approximation (GGTWreP) framework is introduced as an algorithmic means of modeling tactile skills. It connects task goals with interaction control through a process-informed multi-layered structure and forms the basis for most experiments in this thesis. Furthermore, a versatile software stack MIOS has been developed that implements the theoretical foundations.

**Contribution 2: Skill Synthesis Pipeline** A novel process-informed skill taxonomy is introduced that systematically connects manufacturing processes and tactile skills. It encodes robot expert knowledge and connects to learning algorithms so that laypeople such as technicians and shop-floor workers will be enabled to set up automation solutions without robotics knowledge. A synthesis pipeline selects a tactile policy based on formal process descriptions. The GTWreP framework then takes the selected policy and integrates it with the boundary conditions (e.g. success and error conditions) from the process description into a tactile skill model. In further validation experiments, a meaningful number of skills has been implemented and optimized, demonstrating the desired



**Figure 1.1:** The process descriptions that originate e.g. from human technicians are used to synthesize tactile skills based on a taxonomy. The skills are formulated as a learning problem and then optimized by an autonomous learning architecture. Finally, the optimized skills are used in an assembly planning system that solves problems provided by the technician.

high robustness and performance also for challenging real-world manufacturing processes. A collaborative assembly planner is introduced that finds an optimal plan to solve an assembly with a team of humans and robots based on validated tactile skills.

**Contribution 3: Skill Learning** The skill model generated from the synthesis pipeline is formulated as a parameter learning problem, which can then be solved by a state-of-the-art compatible learning algorithm. Experiments demonstrate a very short learning time even for very difficult processes, e.g., such as insertion. During these experiments a transfer learning effect was observed and was investigated through further large-scale experiments. The results indicate strong transfer capabilities of the GGTWreP framework which further improves the versatility of the overall end-to-end approach. The learning and achieved manipulation performance of the skills were directly compared to human capabilities in a reference experiment setup. The results indicate comparable performance for a subset of the implemented skills, and provide insights on the still-existing gaps.

With these fundamental contributions and to the best of the author’s knowledge an unprecedented level of sophistication, performance and resource efficiency in terms of energy and computation has been achieved in robot learning. This constitutes a major step forward in making robots autonomous and learning-enabled as well as able to systematically leverage existing and well established process knowledge.



## 1.4 State of the Art

The following literature overview covers the most relevant topics related to this thesis, i.e. robot platforms, interaction control, skill models, taxonomies, skill learning, and assembly planning. This chapter was written based on [5, 14, 23–25]

### 1.4.1 Robot Platforms

Although there is no hardware development or design in this thesis, the choice of the used robot platform was important in terms of its manipulation capabilities and available low-level access. Most of the experiments were completed with the Franka Emika Robot arm [5, 26], which is the current state-of-the-art robot for compliant physical interaction. The most important properties for its use in this work were its sensitivity, redundancy and kinematic versatility as well as its high-performance research interface that allows for a high degree of freedom when implementing custom controllers. Other arguments were the integrated low-level reflexes [11] and safety-by-design [27, 28]. A thorough introduction to the robot and ways of programming it can be found in [5]. By now the Franka Emika Robot arm has become a widely used platform for robot learning, e.g. [29–32] Furthermore, it has been adopted by the robotics community for various simulation environments, e.g. [29, 33]. Older systems such as the DLR LWR III [1, 8] (which was also used for some preliminary experiments in this work) and the KUKA LWR IV [3] were the first mature systems in this context and started to shift towards joint-side torque control which explicitly uses the dynamics properties of the robot into the control loop. The required sensing of the joint torque is implemented in different ways, e.g. by measuring the current combined with a low gear ratio (e.g. Barrett WAM [34]), by exploiting the implicit compliance of series elastic actuators (e.g. Rethink Robotics Baxter [35] and Sawyer [36]), or by adding torque sensors as e.g. the KUKA iiwa [37] and iisy [38] systems, the Kinova Gen3 arms [4] or the Franka Emika Panda arm. Using joint-torque sensors allows for high-performance compliant control as well as integrated, highly sensitive safety mechanisms such as collision detection with a reaction time of milliseconds.

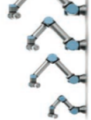

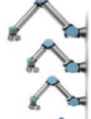
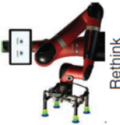




Other systems such as the YuMi from ABB [39] and the Universal Robot arms [40] regulate only position or velocity, neglecting the dynamic characteristics of the joints which leads to very stiff position control. Contact-rich manipulation tasks such as insertion would then require the use of costly F/T sensors. Safe human-robot interaction is in these cases only possible at low velocities and low effective masses.

The PR2 [41] should be mentioned since it has been used in numerous works on robot learning due to its redundant kinematics, large array of sensors, community support and compatibility with numerous ROS packages.

Finally, in Fig. 1.2 an overview of current robot platforms is given regarding their most important properties with respect to their utility in research.

### 1.4.2 Interaction Control

The choice of the right controller is essential for robotics applications and usually goes hand in hand with the choice of the hardware platform. There are controllers that can provide precision in the range of micrometers, and others that can compliantly interact with the environment. For this work, the latter case is much more relevant, since the in-

	 UR Robots	 ABB Yumi	 URe Series	 Reiflink Robotics Sawyer	 Barrett WAM	 KUKA iiwa	 Kinova Gen3	 Franka Emika Robot
<b>Control</b>	Position/Velocity Control	Position/Velocity Control	Position/Velocity Control + Admittance Control	Joint-Torque Control	Joint-Torque Control	Joint-Torque Control	Reflex and Joint-Torque Control	Reflex and Joint-Torque Control
Coordination	N/A	N/A	N/A	Impedance Control	Impedance Control	Impedance Control	Reflex and Impedance Control	Reflex and Impedance Control
<b>ML</b>				N/A	N/A	N/A	Learning Basic Assembly Skills	Learning Basic Assembly Skills
<b>Force Sensing</b>		Current + High Gear Ratio	Current + High Gear Ratio + Force-Torque Sensor	Series Elastic Actuators	Current + Low Gear Ratio	Current + Low Gear Ratio	Strain Gauges	Strain Gauges
<b>Low-Level Interfaces</b>		Position/Velocity	Position/Velocity + Force				Position/Velocity, Joint Torque, Force, Collision Behavior	Position/Velocity, Joint Torque, Force, Collision Behavior
<b>Use Cases</b>	Pick and Place, Machine Tending, Teaching With Pendant	Pick and Place, Machine Tending, Teaching With Pendant	Pick and Place, Machine Tending, Teaching With Pendant + Insertion, Polishing		Compliant Teaching, Compliant Insertions, Polishing, and Grinding	Pick and Place, Machine Tending, Compliant Insertions, Polishing, and Grinding	Sensitive Pick and Place, Sensitive Testing, Force Profile Tracking, Robust Fault Detection, Polishing, Grinding, Complex Sensitive Insertions, Screwing, Drilling, and Compliant Teaching	Sensitive Pick and Place, Sensitive Testing, Force Profile Tracking, Robust Fault Detection, Polishing, Grinding, Complex Sensitive Insertions, Screwing, Drilling, and Compliant Teaching

**Figure 1.2:** A technological overview of the most relevant currently available robot manipulators, their control paradigms, sensors, interfaces, and target use cases. The robot images are taken from [5, 42–47]. N/A: not applicable; UR: Universal Robot

vestigated tactile manipulation skills interact with the environment in potentially complex ways. In the following the rigid body dynamics

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_d + \boldsymbol{\tau}_{\text{ext}}, \quad (1.1)$$

is considered, where  $\mathbf{M}(\mathbf{q})$  is the symmetric, positive definite mass matrix,  $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$  the Coriolis matrix,  $\mathbf{g}(\mathbf{q})$  the gravity vector,  $\boldsymbol{\tau}_{\text{ext}}$  the vector of external link-side joint torques and  $\boldsymbol{\tau}_d$  a controller joint-torque command.

### 1.4.2.1 Impedance Control

Impedance control aims to control the motion of a robot as well as its interaction behavior with the environment. A typical formulation for a rigid,  $n$ -DOF manipulator in Cartesian space is given by

$$\boldsymbol{\tau}_d = \mathbf{J}_x(\mathbf{q})^T(\mathbf{K}_x\mathbf{e} + \mathbf{D}_x\dot{\mathbf{e}}) + \boldsymbol{\tau}_r \quad (1.2)$$

where  $\mathbf{K}_x$  is the Cartesian stiffness matrix,  $\mathbf{D}_x$  is the damping matrix and  $\mathbf{J}_x(\mathbf{q})$  is the Jacobian. The position and velocity errors are denoted by  $\mathbf{e} = \mathbf{x}^* - \mathbf{x}$  and  $\dot{\mathbf{e}} = \dot{\mathbf{x}}^* - \dot{\mathbf{x}}$ , respectively. The dynamics compensator  $\boldsymbol{\tau}_r(t)$  can be defined in multiple ways, see for example [48]. The Cartesian damping matrix requires design e.g. according to [49].

Note that

$$\mathbf{K}\mathbf{e} + \mathbf{D}\dot{\mathbf{e}} + [M_x\ddot{\mathbf{e}}] \quad (1.3)$$

defines the desired interaction dynamics of the robot with the environment. The last term would determine the desired inertia, however, due to the difficulty of measuring or estimating the acceleration in real-world systems, it is usually left out. Note though that this is a topic of current research efforts [50].

Impedance control was first introduced to robotics in [6, 51, 52]. Since then much work has been done to explore the numerous possibilities and variations of this control scheme. In [53] force information was used in addition to improve impedance control for uncertain environment as in deburring, grinding, and assembly tasks. [54] showcases an impedance controller implementation which is able to switch between free motion and a contact task, without having to use inverse kinematic computations. In [55] an analysis of stability properties is provided focusing on two main implementations of impedance control. In [56] an object-centric impedance controller was introduced, which compensates the system's dynamics and directly controls the internal forces of a grasped object.

Especially the Cartesian space formulation of impedance control has been extensively researched, e.g. in [7] where impedance control is compared to admittance and stiffness control and a new impedance controller enhanced by local stiffness control was proposed. In [49] several practical aspects of impedance control are discussed such as nullspace stiffness for redundant manipulators, avoidance of mass matrix decoupling and damping design. [57] introduces a force-tracking impedance controller able to track a desired force in the presence of uncertainties in the environment as well as in the robot dynamic model. The results were demonstrated in simulation and real-world experiments. In [58] impedance control has been applied to 6-DOF industrial robots in several experiments. In [59] a velocity based variable impedance controller is tested for human-robot cooperation using force differentiation to determine human intention. The authors of [8] describe a general passivity-based framework for the control of flexible joint robots that incorporates position, torque and impedance control. In [60] two impedance controllers for

flexible joint robots are proposed based on an inner torque feedback loop. [61] provides a thorough overview of Cartesian impedance control for redundant and flexible-joint robots. In [62] an application of reinforcement learning for learning variable impedance control is demonstrated in simulation and real-world experiments. Current surveys on impedance control can be found in [63,64].

Although single-arm manipulators have been one of the driving factors in impedance control research, the scheme has been successfully applied to various other fields. In [65] applications to lower-limb rehabilitation can be found. Also for hydraulic [66,67] and pneumatic [68,69] actuators impedance control schemes have been developed. Other areas where impedance control has assumed a central role are whole-body control for wheeled [70–72] and biped humanoids [67,73,74], UAVs [75], exoskeletons [76,77], telepresence [78–80], and multi-fingered hands [81].

### 1.4.2.2 Adaptive Impedance Control

Early work on adaptive impedance control can be found e.g. in [82,83]. The authors of [84] introduced a variable impedance controller for cooperation tasks between robots and humans. They first analyzed the human behavior in a human-human tasks and then encoded the results in their controller model. In [85] an adaptive impedance controller was introduced which consists of a filter component, an adaptive controller, and an algorithm that maps the Cartesian-space control input to the joint-space control torque. The controller does not require knowledge of the robot’s dynamics parameters or the inverse kinematics. In [86] model reference adaptive control was applied to impedance control and demonstrated on an industrial robot. The authors of [87] applied an iterative learning scheme to impedance control on a SCARA robot. [88] introduces, tests and compares for nonlinear adaptive impedance controllers for human-robot interaction. Inspired by human motor control [89], the impedance control methodology has been extended to adaptive versions where the stiffness (and possibly damping) and an additional feedforward wrench are dependent on the tracking error [48,90,91]. The exact behavior is governed by a learning factor and a forgetting factor. Such an adaptive impedance controller is given by

$$\boldsymbol{\tau}_d = \mathbf{J}_x(\mathbf{q})(-\mathbf{f}_{ff}(t) - \mathbf{f}_d(t) - \mathbf{K}_x(t)\mathbf{e} - \mathbf{D}_x[\mathbf{M}(\mathbf{q}), \mathbf{K}(t, \mathbf{K})]\dot{\mathbf{e}}) \quad (1.4)$$

where  $\mathbf{f}_{ff}(t)$  is the adaptive feedforward wrench,  $\mathbf{f}_d(t)$  is an additional predefined wrench trajectory that encodes potential prior knowledge, and  $\mathbf{K}_x(t)$  is the adaptive stiffness matrix. The approach was also extended to a deformable environment [9].

### 1.4.2.3 Force Control

A typical force controller in Cartesian space can be expressed by

$$\boldsymbol{\tau}_d = \mathbf{J}_x(\mathbf{q})^T(\mathbf{k}_p\mathbf{f}_e + \mathbf{k}_i \int_0^t (\mathbf{f}_d(\sigma) - \mathbf{f}_{\text{ext}}(\sigma))d\sigma + \mathbf{k}_d\dot{\mathbf{f}}_e) \quad (1.5)$$

where  $\mathbf{k}_p$  is the proportional gain,  $\mathbf{k}_i$  is the integral gain,  $\mathbf{k}_d$  is the derivative gain of the control law, and  $\mathbf{f}_e = \mathbf{f}_d - \mathbf{f}_{\text{ext}}$  is the force error.

Force control is e.g. used in unmodeled contact situations or if the goal of a task requires precise (even dynamic) force tracking.

Force control in robotics has been researched for many years now. In [92] a method for combining motion control and force control has been presented based on the operational space formulation. [93] provides an early (historical) review of force control in robotics. The authors of [94, 95] provide insights into the role of dynamics and nonlinearities in force control approaches. In [96] a force control scheme without force sensor based on a disturbance observer is presented and demonstrated in a real-world experiment. [97] introduces an adaptive force control algorithm for velocity and position controlled robot arms. Force control has been applied in many robotics-related areas. The authors of [98] presented an observer-supported force controller that is able to compensate the motion of a beating heart, thus aiding cardiac interventions. In [99] a controller to track desired contact forces for whole-body approaches in humanoids is presented. Other research focuses on contact forces in human-robot collaborations [100] or surface electromyogram-based control strategies for exoskeletons [101]. Most force controller schemes rely on external force/torque sensors or joint torque sensors and a momentum observer [102]. Others have also shown approaches without force sensing [103]. Some earlier surveys can be found in [104–106].

#### 1.4.2.4 Unified Force / Impedance Control

Unified force / impedance control combines the advantages of force control and impedance control in a single controller. Early work can be found in [107] that enables impedance control and position-limited force control simultaneously. [108] introduces a unified controller connected to an energy tank that preserves passivity of the system. This approach was extended in [109]. [110] presents a force / impedance control augmented with a kinestatic filter, which ensures that the commanded pose and wrench are consistent with a given task model. [111] proposes a unified motion / force / impedance controller for unknown contacts.

A unified force/impedance controller can be written as

$$\boldsymbol{\tau}_d = -\mathbf{J}_x \left[ \mathbf{k}_p(\mathbf{f}_d - \mathbf{f}_{\text{ext}}) + \mathbf{k}_d(\dot{\mathbf{f}}_d - \dot{\mathbf{f}}_{\text{ext}}) + \mathbf{k}_i \mathbf{h}_i(\mathbf{f}_{\text{ext}}, t) + \mathbf{K}_x \mathbf{e} + \mathbf{D}_x \dot{\mathbf{e}} \right] \quad (1.6)$$

where  $\mathbf{K}_x$  is the Cartesian stiffness matrix,  $\mathbf{D}_x$  is the Cartesian damping matrix,  $\mathbf{k}_p$ ,  $\mathbf{k}_d$ , and  $\mathbf{k}_i$  are the force controller gains, and  $\mathbf{h}_i$  is defined as

$$\mathbf{h}_i(\mathbf{f}_{\text{ext}}, t) = \int_0^t (\mathbf{f}_d(\sigma) - \mathbf{f}_{\text{ext}}(\sigma)) d\sigma \quad (1.7)$$

### 1.4.3 Skill Frameworks

This section presents the state of the art in skill frameworks. The first part focuses on skill model structures, and the second part on policy representation.

Note that in the literature the terms skill, task, action, capabilities and others are often used to describe the same thing. In this thesis the terms skill and task are used. A task  $\iota$  is defined as a concrete manipulation problem, i.e. it is unique. A skill  $\varsigma$  is the means to solve the task, but it is not restricted to solve only that particular task. However, it is valid to say that a skill is learned, despite the fact that the robot interacts only with one task. Thus, skill and task have a similar meaning when used in the context of learning.

### 1.4.3.1 Skill Models

There has been much work on representing robot manipulation skills or actions, yet no common representation has been established so far. In fact, in most cases there is no formal frame for manipulation skills since researchers and engineers tend to create their own custom solutions for most approaches. In [112] the authors thoroughly reviewed the current research in this area and came to the conclusion that a common basis for representing skills is missing, although there are numerous works that each partially cover the same or similar aspects and may finally lead to a unifying formalism. Specifically, they point several open research challenges regarding skill representations, i.e. effect grounding, couple forward and inverse models, exploiting language for action understanding, intrinsically motivated learning, selective attention, the correspondence problem, and sequence-based modeling.

Early work on skill representations and frameworks can be found e.g. in [113] where an intermediate level between incoming sensor data and the symbolic level acts as a bridge, [114, 115] where AND/OR graphs are used to represent assembly plans and connect to a planning layer, or [116] where hidden Markov models are used to represent observed human actions. The authors of [117] describe a hierarchical abstract behavior architecture combining hierarchical task structures with behavior-based systems. Their approach enables reuse of simple behaviors across multiple tasks suggesting a notion of atomic actions. It was experimentally validated on a mobile robot. [118] provides an approach inspired by neuroscientific and psychological data. The authors describe a cognitive architecture designed for action recognition and imitation using a distributed system of inverse and forward models. [119] proposes a bio-inspired process of imitation learning using goal-directed action sequences consisting of motor primitives. In [120] a hierarchical task representation using primitive action sequences is proposed. They are grounded by using human demonstrations. In [121] an unsupervised learning approach for action primitives based on human demonstration is proposed. The primitives are represented by parametric hidden Markov models which encode motion trajectories for a robot. The authors discuss how such primitives can be learned, how they can be used to synthesize movements to achieve desired effects, and how the model can be used to detect action primitives and the entailing effect from observing human demonstrations. In [122] a high-level reasoning framework was developed using DMPs as low-level policy.

Sometimes, researchers adapt existing formal methods to robot manipulation such as Petri nets for higher-level components [123–126] and behavior trees [127–130] which are also often used to represent control policies to a certain degree.

In [131], object action complexes (OAC), one of the first attempts at bridging low-level control and high-level planning and learning, are introduced. In [132, 133] OACs are connected to a robot vision system to segment and extract properties of objects. The authors argue that objects and actions are inherently coupled. In [134, 135] a formal definition and examples of OACs are provided. The authors of [136] demonstrate the application of OACs to a programming by demonstration approach to create sequences of actions for a household kitchen scenario.

The authors of [137] have developed a multi-layered architecture consisting of a semantic event chain (high-level), a finite state machine (mid-level), and a hardware level (low-level). On the high level actions are described in a symbolic and abstract way by relating them to relevant objects. The mid level utilizes a finite state machine to execute low-

level action primitives in the correct sequence. This approach is tested on a variety of real-world manipulation problems.

There are other approaches based on a strong link between actions and objects. For example, in [138] actions and object representations are used to build generative models for internal simulations on action outcomes, in [139] representations for reasoning about the effects on objects due to actions are developed, and in [140] a knowledge base for action and object representations is introduced.

The authors of [141] have proposed an assembly skill framework which models skills based on trajectories described in pose-wrench space as well as a finite state machine. The approach has been tested in a real-world experiment involving a snap-fit problem.

[142] introduces a planning and execution framework called SkillMaN which combines learned or user-provided knowledge on skills with geometric reasoning and task planning. Many researchers combined the concepts of language and action due to their inherent similarity. [143] discusses how actions and language can be represented and combined and suggests a number of research milestones and test scenarios for cognitive robotics.

### 1.4.3.2 Policy Representations

Manipulation policies express a solution to a manipulation problem in a formal, quantitative way. There are many ways how such policies can be represented and most of them can be separated into three classes (loosely based on (Kroemer et al. 2019)):

- **Non-parametric policies:** This is the most general and expressive type of policy. Although they are very flexible, these policies require large amounts of data. Prominent examples are:
  - **Locally-weighted regression:** Locally-weighted regression is a way of estimating a regression surface using a multivariate smoothing procedure. A function of the independent variables is fitted locally and based on a similar mechanism how a moving average for a time series is computed. In [144] a survey on locally weighted learning can be found, in [145] an application to learning high-dimensional problem spaces is shown. In [146] locally weighted learning is applied to various high-dimensional robotics problems such as pole-balancing and inverse-dynamics learning.
  - **Gaussian processes:** A Gaussian process is a stochastic process such that every finite linear combinations of the contained random variables is normally distributed. They provide a probabilistic non-parametric modeling approach for black-box identification problems for non-linear dynamic systems. In [147], applications to model-based predictive control. The authors of [148] have used Gaussian processes to model robot tasks in a learning-by-demonstration approach. In [149] an application of Gaussian processes to learning the inverse dynamics of a robot. In [150] Gaussian processes have been utilized to control the high-dimensional space of a simulated octopus arm.
- **Generic fixed-size parametric Policies:** This type of policy usually makes stronger assumptions about the structure of the policy. The representational capabilities are still very broad, however, much more design choices are involved that determine the frame. Examples are:

- Look-up tables: Usually, a look-up table is a data structure (e.g. an array or a hash map) which allows for quick access to data based on a key, since no additional computation is required. It is not often used as a policy representation, but a few applications are shown in [151].
- Fourier basis: The Fourier basis is based on the Fourier series and used for linear function approximation. Applications for reinforcement learning can be found in [152, 153].
- Neural networks: Neural networks are one of the most widely used and popular policy representations today and are applied to a wide spectrum of domains such as image and object detection, speech recognition and robotics. They consist of multiple layers of artificial neurons containing specific parameterized activation functions such as rectified linear units, sigmoid functions or softmax. For example, neural networks have been used to learn basic manipulation tasks using a mapping from the robot state [20] and visual input [154] to robot torques.
- Restricted parametric (goal-driven) policies: These are specialized policy representations with limited representational capabilities. However, these types of policies usually exploit existing structures of the problems they are applied to. This exploitation is mostly based in their design and requires manual choices. In turn, their complexity is reduced which allows for much more sample-efficient learning. Examples are:
  - Structured neural network architectures: These are neural networks with an inherent structure that allows for higher performance on specific problem domains. Examples are [155] which introduces a generalized value iteration network that is evaluated on a range of benchmark problems, and [156] where the authors describe a network based on first-order principles for inverse dynamics learning.
  - Dynamic motion primitives: Dynamic motion primitives (DMP) [146, 157, 158] consist of a nonlinear attractor dynamics and an additional oscillatory term. DMPs have been shown to be able to generate human-like motions and are also capable of handling contacts with the environment. They are often used for programming by demonstration approaches [159–161]. In [162], the authors introduce a method to sequence DMPs and demonstrate it in a real-world experiment with tasks such as pouring and table wiping. In [163] DMPs have been used in combination with potential fields for obstacle avoidance. Other applications of DMPs to physical manipulation include [164], where a multi-fingered hand folded a piece of paper, [165] where DMPs were part of a framework including object detection, pose estimation and manipulation for grasping of everyday objects by a humanoid robot, and [166] where a bimanual collaborative human-robot task was performed using DMPs.
  - Linear Quadratic Regulators and Linear Quadratic Gaussian based methods: These methods attempt to describe the dynamics of a system by a set of linear quadratic equations. In [20] an example to robot manipulation learning can be found, in [167] for planning grasping tasks.



### 1.4.4 Skill Taxonomies

In general, a taxonomy is a systematic approach to classify things in groups or types in a structured way and can significantly aid in providing further insights into the respective field often allowing for a wider perspective. Many taxonomies have a hierarchical tree structure, however, this is not a requirement. The most commonly known and oldest taxonomies are the classifications of organisms [168] which are constantly updated and revised as new information becomes available. These taxonomies classify e.g. animals according to their relation to each other, nowadays also supported by DNA sequencing. Other prominent examples are the classification of diseases [169], which orders diseases according to their cause, pathogenesis or symptoms, of viruses [170] where viruses are classified by e.g. their morphology and nucleic acid type, and folk taxonomies [171] which order e.g. animals and plants according to social knowledge and everyday language embedded in specific cultures. In the economic domain corporate taxonomies hierarchically classify physical or conceptual entities such as products, processes, documents, or job titles [172]. The human factors and classification system [173] is a safety taxonomy designed to identify the human cause of accidents, which was primarily motivated by the rate of human error related to flight accidents.

The taxonomy introduced in this thesis is concerned with manipulation skills for autonomously acting robots. In the following related work in this area is reviewed and followed with a review on taxonomies in robotics in general.

#### 1.4.4.1 Skill Taxonomies

Extensive real-world-tested manipulation skill taxonomies are difficult to build since they require a significant effort in terms of programming, experimentation and validation. In [174] the authors describe a taxonomy of haptic tasks based on human manipulation which also relates to robotics. First, they categorize manipulation action by broad labels such as significant required arm strength or the need of two-handed manipulation. Then, they added a second category that is determined by the direction of force and/or torque used in the respective task. In [175] an assembly taxonomy is proposed which represents the decomposition of complex assembly tasks into simple skills which can be reused in order to reduce programming time and overhead. Another approach is shown in [176]. The authors combine high-level AI-planning methods and compliant manipulation schemes in order to classify typical household chores. They created a hierarchical taxonomic structure that classifies manipulation tasks with respect to contact complexity. Their taxonomy makes use of general contact classifiers to make it accessible to high-level planning and introduces a sub-categorization based on the detailed contact situation to further distinguish manipulation actions. There has also been work that aims more at the integration of robots into established frameworks. In [177] the authors discuss the compromise between standards and flexible formalisms in industry. Although they do not make use of a taxonomic approach they argue for the use of flexible skills and their connection to context-dependent data. In [178] skills are derived from an abstract process view and defined within the concept of PPR in AutomationML. Skills are understood as more general abilities and tasks as concrete instantiations. Although not directly a taxonomy of manipulation skills, [179] introduces a system of benchmark tasks categorized by the type of test and the tested part of the robot. They furthermore provide guidelines for designing test protocols and experiments. In [180] a taxonomy of motions related to

cooking tasks is developed. The motions are grouped into similar sets based on trajectory and contact attributes.

#### 1.4.4.2 Other Taxonomies in Robotics

Among the most prominent works on taxonomic structures in robotics and related fields are grasp taxonomies. Those taxonomies do not directly relate to the work in this paper but provide an important source of inspiration in terms of structure. They usually classify different grasps in a hierarchical structure for human or anthropomorphic hands. Most notably among them are the classifications of Cutkosky [181] and Bullock [182]. In [183] these taxonomies are extended by classification elements related to tactile interaction. One of the first approaches towards such a taxonomy is [184] that divided grasps into power and precision grasps. In [185] a taxonomy is presented which contains various approaches to structuring swarm robots. In the field of multi-robot system, the authors of [186] proposed a taxonomy of task allocation where it is shown that many problem related to multi-robot task allocation can be viewed as instances of other, well studied, optimization problems. It is demonstrated how the taxonomy can be used to synthesize new approaches based on existing theory. This taxonomy is extended by further research in [187]. Taxonomic research has also been done e.g. in humanoid robotics. In [188, 189] a taxonomy for whole-body poses and the transitions between them has been developed. The authors propose a formal definition to characterize whole-body poses and to provide a framework for planning and benchmarking whole-body motion and loco-manipulation. In [190] a taxonomy of robot-assisted training systems is introduced that classifies existing systems by e.g. requirements, interaction types, level of autonomy and possibilities of personalization. One result of this taxonomy is a set of open challenges for designing and developing such training systems. A more general taxonomy of robotic hardware systems is proposed in [191]. The authors developed a representation of robots that links application requirements to robot capabilities with the aim of aiding application developers in choosing the right platform for their purpose. [192] introduced a more specific taxonomy of socially interactive robots.

A taxonomy of action representations in robotics is given in [112]. In [193] a taxonomic overview of programming by demonstration approaches is developed. The authors reviewed existing work and identified the basic elements of current methods. In the field of human-robot interaction [194] introduced a taxonomy that classifies human-robot interaction scenarios by e.g. the role of the human, communication channels, proximity or field of application.

#### 1.4.5 Skill Learning

Machine learning (ML) is a topic that is entangled with robotics for quite some time now since already 50 years ago first ideas were formulated [195] and experiments were conducted. Early surveys can be found in [144, 196–200], more recent ones in [201–210]. Although machine learning in general as well as in the context of robotics is a very broad field, in this section the focus explicitly lies on manipulation learning and related topics. In general, machine learning is often separated into supervised learning, unsupervised learning and reinforcement learning. Although, the focus of this work is on supervised learning. However, reinforcement learning is of significant importance to the field of robot

manipulation learning [206], thus related work from this field is also reviewed. Although used in robotics for some applications [211–214], unsupervised learning is not relevant for the present use case and is therefore omitted.

Since this thesis focuses on real-world manipulation learning the following review is divided into simulation-based and real-world-based learning. Additionally, the special case of transfer learning is reviewed which also covers learning from demonstration and sim-to-real approaches. Finally, overview of algorithms used in robot manipulation learning is given.

#### 1.4.5.1 Manipulation Learning in Simulation

Learning robot manipulation in simulation is a widely-spread practice for benchmarking learning methods and to avoid the costly and time-demanding setup of real-world systems. However, the gap between simulation and reality is, depending on the task, significant. Whereas simple position-based tasks such as moving, grasping and pushing objects can be realized in simulation, more contact-intensive interactions such as insertion or bending objects prove to be much more difficult in the real world. Nonetheless, the work done in simulation has provided the research community with much needed guidance in terms of algorithms and general experience with manipulation learning. Furthermore, sim-to-real is a term coined by the effort of transferring results acquired in simulation to the real world with no or only minimal additional learning. This specific branch of robot learning is reviewed in Sec. 1.4.5.3.

Early work in this field can be found in [215] where reinforcement learning is used in a hierarchical context. First elementary skills are learned and then high-level coordination policies. The approach is demonstrated for a simulated mobile robot.

Due to their straight-forward usability and reliance on computers alone Simulations have been leveraged to create various benchmark and test suites, especially for reinforcement learning approaches. Examples are SURREAL [216] which is an open-source framework offering various benchmark manipulation or RL Bench [29] which also offers a wide variety of tasks in terms of complexity. In [217] a set of simple pushing and pulling tasks were learned on different simulated robots. The authors of [218] propose an approach based on Gaussian mixture models to learn and reproduce manipulability ellipsoids from expert demonstrations.

#### 1.4.5.2 Manipulation Learning in the Lab

Learning high-performance real-world robot manipulation is one of the most intriguing and important challenges in robotics research. Solving it would allow for completely different approaches to automation in industry, meaning faster setup times and lower energy consumption. However, to this date there has been no real application of autonomous robot learning to relevant, real-world problems in an industrial, commercial scenario. The jump out of the lab has not yet happened. The reasons are mostly robustness, task difficulty and lack in general autonomy of the robotic systems. Despite the still considerable obstacles, the research community has had impressive progress regarding real-world robot manipulation learning. In [219] learning of two motor skills, i.e. ball-in-a-cup and ball padding were learned on a real Barret WAM robot using both imitation and reinforcement learning. In [220] the authors demonstrated learning a pool stroke and a box flipping task

on a PR2 dual-arm robot. In [221] a survey of applications and real-world challenges for reinforcement learning in robotics is given. In [222] hierarchical learning of complex skills has been explored. The robot learns a probabilistic model of the various phases and phase transitions of a skill from human demonstrations. Basic motor primitives are used to transition between the phases. [223] demonstrates the usage of Gaussian mixture models for robot manipulation in both simulation and real-world experiments. The approach uses synergistic basis vectors tied to the covariance matrices of the Gaussian mixture model to allow for reuse of policy parts. [20, 154] showcases the learning of several manipulation tasks such as closing a bottle or stacking Lego blocks based on an end-to-end approach which maps the robot state and additional visual input to robot torques. This approach has found some popularity and was applied to other real-world tasks such as opening a door [21], moving objects [224] and long-term tasks involving grasping and placing objects [225]. In [226] guided policy search is extended to be used for random initial states and a reset-free procedure, and demonstrated in simulation and a real-world placing task. The authors of [227] demonstrate the application of soft Q-learning to real-world manipulation tasks and showcase a higher sample efficiency than model-free approaches typically used in reinforcement learning. In [228] the application of generative motor reflexes to a real-world peg-in-hole task was demonstrated.

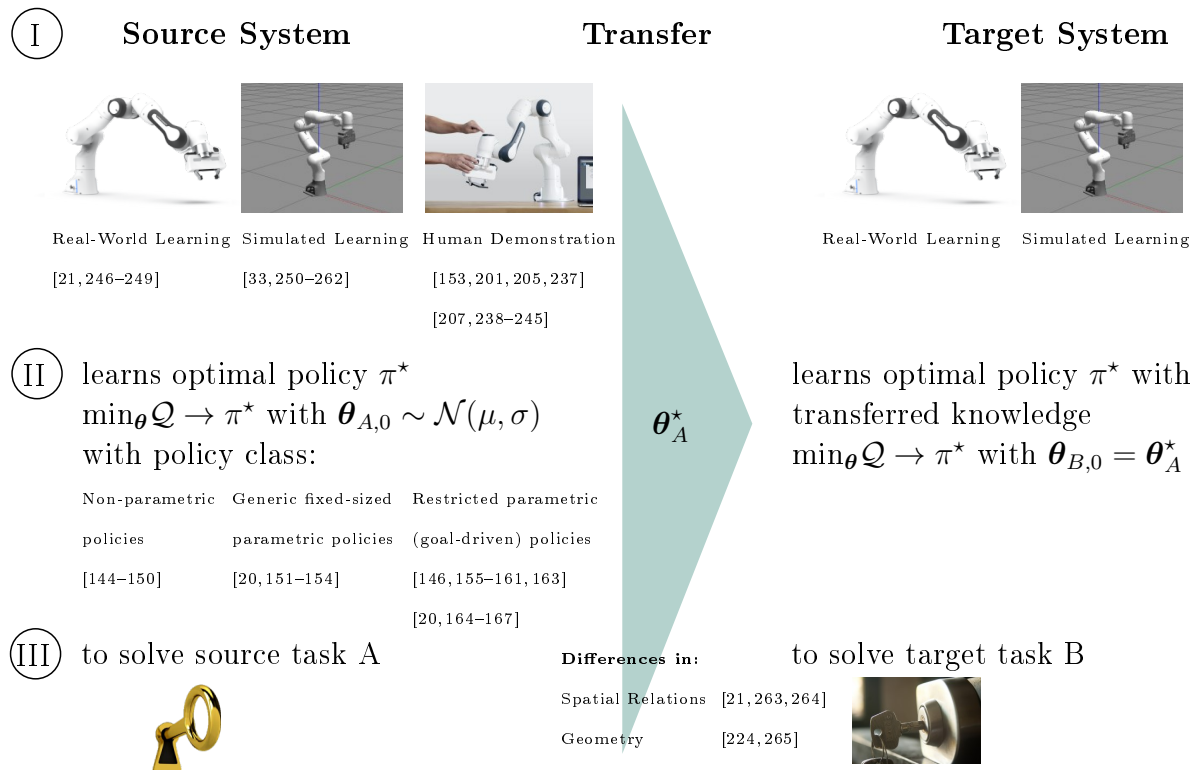
In [229] a deep predictive policy training is proposed for data-efficient skill learning. The approach is demonstrated for a grasping and ball throwing task on a real robot. Deep reinforcement learning has in some cases also been applied to very difficult assembly problems as shown in [230] where an industrial-grade insertion problem with very tolerances has been learned.

The authors of [231] explicitly look at possible pitfalls and difficulties when setting up a reinforcement learning experiment for real-world problems. They also suggest some steps to mitigate these challenges.

### 1.4.5.3 Transfer Learning

In general, the process of transferring knowledge is usually referred to as transfer learning, which is described in detail in [232, 233] while good surveys can be found in [234–236]. Interestingly most work on transfer learning and related topics can be found on classification, language processing and computer vision, and only few researchers address physical robotic applications, such as transfer between different manipulation skills and robot systems [217]. Moreover, in real-world robot applications one has to distinguish between mostly position-based problems (such as grasping, pick-and-place or positioning) and interaction-based problems with complex and hard-to-observe dynamics (such as insertion). The former is much easier to simulate (and transfer to the real world) since it is essentially a vision problem. Literature about transferring learned experience in robotics alone is filled with numerous different approaches. In order to give an overview of existing approaches, they are reviewed in terms of the source system and the variations between source and target tasks (such as goal state, cost function, environment etc.). Furthermore, only cases where the target system is a real-world robot are considered.

Related works are categorized based on their respective i.) source and target system combination, ii.) underlying model of the manipulation skill policy, iii.) and differences between the source and target tasks such as goal state, environment, cost function, etc. (see Fig. 1.3). Note that policies are reviewed in Sec. 1.4.3.



**Figure 1.3:** Categorization of transfer learning based on the source and target systems, policy representation, and source and target tasks.  $\pi^*$  denotes an optimal policy,  $\mathcal{Q}$  a suitable quality metric, and  $\theta$  a set of parameters that define the policy  $\pi$ . The source system, policy, task space changes, and related work are all categorized.

## Source System

The source system denotes the origin of the transferred knowledge or information, i.e., by which system the source skill was learned. Three basic cases are distinguished:

- **Human demonstration:** Learning by (human) demonstration (LbD) is a well-researched topic that describes the process of a robot learning a skill by observing a human teacher [153, 201, 205, 207, 237]. The employed methods range from direct kinesthetic interaction [238, 239], tracking and imitating human kinematics [240], to analyzing video recordings of humans [241]. Other source technologies, such as EMG signals, have also been used in [242] various applications, including navigation [243], household assistance, [244] and cleaning [245]. LbD assumes that robots and humans are kinematically similar enough, or that a human is able to compensate the dynamics of a robot arm during a demonstration.
- **Sim-to-real:** Much effort has gone into researching sim-to-real approaches to utilize the advantages of fast and easily scalable simulations in robot learning by transferring the acquired knowledge to the real world [250–254]. [33] used simulations that were informed by real-world roll-outs to learn policies for real-world skills, such as opening a drawer, on two different robots. In [255], a transfer of manipulation policies was shown by first using randomized dynamics in simulation and then in the real world. [256] addressed the problem of learning manipulation skills that involve deformable objects in simulation. This approach implies that the simulated world is accurate enough to forecast real-world processes (or, at least, they can be

suitably informed via model fitting). One example of using transfer learning on parameterized skills is [257], where a simulated robot was able to throw a ball to different target locations. [258] aimed to reduce the amount of fine-tuning when transferring the results to the real world by changing the simulation parameters. Recently, [259] introduced a robust value iteration algorithm to find more robust policies when transferring them from simulation to the real world. They showcased superior robustness compared to state-of-the-art deep reinforcement learning methods. In [260], a simulation-trained approach was able to handle object rearrangement problems. [261] used hybrid sim and real training to enable a pneumatically-driven robot to learn to play table tennis. In [262], an anthropomorphic hand was able to solve a rubiks cube solely based on the perception of the hand position and interaction torques, which were learned using an initial simulation step.

- **Real-world robot:** There have not yet been many examples of using a real robot as a knowledge source. However, a few examples do exist. For instance, [246] transferred simple RL policies for a ball-hitting skill, and [247] transferred learning for the controllers of two quadrotor platforms. Another early work is [21], where multiple robots learned to open (equivalent) doors collaboratively. First methods to share policies between real-world agents were developed, although the differences between the tasks were only on the level of spatial displacements. A different approach is to use real-world data to feed simulations [248]. However, the lack of real-world experiments in literature demonstrates how difficult this challenge is. Partially unknown dynamics or dynamics that have not been considered for the used robot platforms contribute to the complexity of this challenge. Furthermore, the material properties and interaction dynamics of the learned tasks also make it difficult. This implies that standard big-data approaches will ultimately run into problems, since it is not possible to scale up the setups in the same way as in simulated worlds. Domain randomization may help pushing the boundaries of this obstacle [249].

### Skill Variations

On the source system a skill is learned by performing it in a specific task environment. The differences between source and target skill are related to the specific configuration of the environment (kinematics), geometric properties of involved objects, a goal state and typically an explicit or implicit optimization goal.

- **Spatial Relations:** The spatial relations of the target skill may vary in terms of the goal poses or kinematic states of the environmental elements like dynamic objects. Such differences could range from a few millimeters to completely different orientations and large translations. Many works introduced minor variations [263] to analyze the robustness of the learned skill. However, generalization to arbitrary pose shifts for contact-rich manipulation problems (as for contact-free motion problems [264]) were not yet achieved. Another form of spatial difference is transferring between skills that have previously been learned on physically different but geometrically identical setups, e.g., [21].
- **Geometry:** The object geometry may change significantly [224, 265]. This type of change ranges from small variations in, for example, the length or diameter, to completely different object types, such as from a cylinder to a key.

- **Goal State:** The goal state defines the desired final world state after the skill has been executed. This is either expressed implicitly by a loss function and/or is explicitly encoded in the policy for goal-driven approaches. If the goal is changed substantially, e.g., from inserting an object to pushing it, the knowledge transfer process becomes significantly more complicated. At this point, the skill has changed on a more fundamental level compared to, for example, two insertion skills with different objects. To the best of the author’s knowledge, no works have yet explored this level of knowledge transfer.

It is important to note that without any change in the skill configuration, there is, by definition, no new skill to be learned. In that case, learning from human demonstration or a different real robot can be considered a head start, since it is the same skill just with more information about the initial state. Similarly, transferring knowledge from simulation to the real world would be a test of the simulator’s performance and accuracy in the case that also the source and target systems are the same. Scenarios with only small changes in the kinematic configuration may also be counted towards this.

### 1.4.6 Robot Software Frameworks

There exist many robot frameworks, also specifically for learning. An early attempt at an open-source, community-driven framework is orocos [266]. One of the most popular and widespread ones is the Robot Operating System (ROS) [267]. Its principle is to have nodes that communicate in the network with a publisher / subscriber mechanism. It is highly flexible and offers a vast library of user-generated content for various applications and hardware systems. It currently is about to be replaced by its successor, ROS 2 [268] which has similar design principles but has a much higher performance. There are several meta frameworks that build on ROS, such as [269] that provides real-time capabilities, or [270] that adapts ROS for multi-robot systems specifically.

There is a variety of smaller, and usually more specialized, frameworks. ArmarX is designed to program humanoid robots with a rich toolset [271]. [272] introduces XBotCore, a real-time capable robot control framework for EtherCat-based robots and software middleware, that satisfies hard real-time requirements. [273] presents the KnowRob2 framework that is designed for knowledge processing and reasoning.

### 1.4.7 Assembly Planning

Human-robot collaboration [274–277] has developed into a large field involving various research directions such as human-robot interaction [278, 279], safety [280–282], and (multi-agent) task planning and allocation [283–287]. Assembly planning is highly related to human-robot collaboration since human workers and robots are often utilized as a team. Numerous works have built entire frameworks to solve assembly problems. [288] provides an overview on human-robot collaborative assembly and the used methods. To represent assembly planning problems on task level in robotics, various approaches have been used such as AND/OR graphs [114] and Petri nets [289]. Other works focus the planning on motion level such as [290] that presents an approach that solves an assembly problem on motion level through the use of a genetic algorithm. The ROBO-PARTNER project’s aim was to develop intelligent assembly cells, task planning, and communication methods for

collaborative human-robot assembly [291]. [292] shows a multi-layered framework called flexHRC which also uses AND/OR graphs to model human-robot cooperation models with an additional focus on the use of wearable sensors for additional data on the environment and human co-workers. [293] introduces a cyber-physical assembly system-based metaheuristic for automated assembly sequence planning. [294] developed a method to transfer knowledge about constraints from one assembly onto another.

## 1.5 Thesis Structure

The thesis is structured as follows. Chapter 2 provides the theoretical foundations for tactile skills, their synthesis from formal process definitions, the taxonomy of manipulation skills, skill learning, and planning. Chapter 3 introduces the GGTWreP framework and its implementation, the Machine Intelligence Operating System (MIOS). Furthermore, it presents numerous validation experiments such as related publications and demonstrators shown at public events that verify the overall learning architecture. In Ch. 4 the theoretical foundations are experimentally validated using the architecture. It presents experiments and results from skill synthesis, skill learning, transfer learning, and assembly planning. Finally, Ch. 5 concludes the thesis and gives an outlook on future work.

## 1.6 Curriculum Vitae

### Education

Since 04/2018	<b>Dissertation</b> Technical University Munich
09/2014–03/2018	<b>Dissertation</b> Gottfried Wilhelm Leibniz Universität Hannover
10/2012–07/2014	<b>Master of Science</b> Gottfried Wilhelm Leibniz Universität Hannover
10/2007–09/2012	<b>Bachelor of Science</b> Gottfried Wilhelm Leibniz Universität Hannover
08/2003–07/2007	<b>High School Diploma</b> Fachgymnasium Wirtschaft (Wunstorf)

### Professional Experience

Since 04/2022	<b>Franka Emika GmbH</b> Head of Robotics Learning
04/2018–03/2022	<b>Research Assistant</b> Technical University Munich
09/2014–03/2018	<b>Research Assistant</b> Gottfried Wilhelm Leibniz Universität Hannover



## KPI Statistics

Citations	857
h-index	11
i10-index	11

July 25, 2024

## Publications

### Conference Papers

- **Johannsmeier, L.**, & Haddadin, S. (2022, October). Can we reach human expert programming performance? A tactile manipulation case study in learning time and task performance. In Proc. International Conference on Intelligent Robots and Systems (IROS) (pp. 12081-12088). IEEE.
- Chen, X., **Johannsmeier, L.**, Sadeghian, H., Shahriari, E., Danneberg, M., Nicklas, A., ... & Haddadin, S. (2022, October). On the Communication Channel in Bilateral Teleoperation: An Experimental Study for Ethernet, WiFi, LTE and 5G. In Proc. International Conference on Intelligent Robots and Systems (IROS) (pp. 7712-7719). IEEE.
- Voigt, F., **Johannsmeier, L.**, & Haddadin, S. (2020). Multi-Level Structure vs. End-to-End-Learning in High-Performance Tactile Robotic Manipulation. In Proc. Conference on Robot Learning (CoRL) (pp. 2306-2316).
- Grischke, J., **Johannsmeier, L.**, Eich, L., & Haddadin, S. (2019, May). Dentronics: review, first concepts and pilot study of a new application domain for collaborative robots in dental assistance. In Proc. International Conference on Robotics and Automation (ICRA) (pp. 6525-6532). IEEE.
- Kühn, J., Ringwald, J., Schappler, M., **Johannsmeier, L.**, & Haddadin, S. (2019, May). Towards semi-autonomous and soft-robotics enabled upper-limb exopros-thetics: first concepts and robot-based emulation prototype. In Proc. International Conference on Robotics and Automation (ICRA) (pp. 9180-9186). IEEE.
- **Johannsmeier, L.**, Gerchow, M., & Haddadin, S. (2019, May). A framework for robot manipulation: Skill formalism, meta learning and adaptive control. In Proc. International Conference on Robotics and Automation (ICRA) (pp. 5844-5850). IEEE.
- Haddadin, S., & **Johannsmeier, L.** (2018, October). The art of manipulation: Learning to manipulate blindly. In Proc. International Conference on Intelligent Robots and Systems (IROS) (pp. 1-9). IEEE.

- Grassmann, R., **Johannsmeier, L.**, & Haddadin, S. (2018, October). Smooth Point-to-Point Trajectory Planning in  $SE(3)$  with Self-Collision and Joint Constraints Avoidance. In Proc. International Conference on Intelligent Robots and Systems (IROS) (pp. 1-9). IEEE.
- Shahriari, E., **Johannsmeier, L.**, & Haddadin, S. (2018, June). Valve-based virtual energy tanks: A framework to simultaneously passify controls and embed control objectives. In Proc. American Control Conference (ACC) (pp. 3634-3641). IEEE.
- Haddadin, S., **Johannsmeier, L.**, Becker, M., Schappler, M., Lilge, T., Haddadin, S., ... & Parusel, S. (2018, March). roboterfabrik: a pilot to link and unify German robotics education to match industrial and societal demands. In Proc. International Conference on Human-Robot Interaction (pp. 375-375). IEEE.

## Journals

- Ringwald, J., Schneider, S., Chen, L., Knobbe, D., **Johannsmeier, L.**, Swikir, A., & Haddadin, S. (2023). Towards Task-Specific Modular Gripper Fingers: Automatic Production of Fingertip Mechanics. *Robotics and Automation Letters (R-AL)*, 8(3), 1866-1873.
- Haddadin, S., Parusel, S., **Johannsmeier, L.**, Golz, S., Gabl, S., Walch, F., ... & Haddadin, S. (2022). The franka emika robot: A reference platform for robotics research and education. *IEEE Robotics & Automation Magazine (RAM)*, 29(2), 46-64.
- Naceri, A., Elsner, J., Tröbinger, M., Sadeghian, H., **Johannsmeier, L.**, Voigt, F., ... & Haddadin, S. (2022). Tactile Robotic Telemedicine for Safe Remote Diagnostics in Times of Corona: System Design, Feasibility and Usability Study. *Robotics and Automation Letters (R-AL)*, 7(4), 10296-10303.
- Grischke, J., **Johannsmeier, L.**, Eich, L., Griga, L., & Haddadin, S. (2020). Dentronics: Towards robotics and artificial intelligence in dentistry. *Dental Materials*, 36(6), 765-778.
- Shahriari, E., **Johannsmeier, L.**, Jensen, E., & Haddadin, S. (2019). Power flow regulation, adaptation, and learning for intrinsically robust virtual energy tanks. *Robotics and Automation Letters (R-AL)*, 5(1), 211-218.
- Haddadin, S., **Johannsmeier, L.**, & Ledezma, F. D. (2018). Tactile robots as a central embodiment of the tactile internet. *Proceedings of the IEEE*, 107(2), 471-487.
- **Johannsmeier, L.**, & Haddadin, S. (2016). A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes. *Robotics and Automation Letters (R-AL)*, 2(1), 41-48.

### Workshops

- **Johannsmeier, L.**, Ringwald, J., Kühn, J., & Haddadin, S. (2016). Teleoperated semi-autonomous control of the lwr and a humanoid hand via the myo armband. Workshop in International Conference on Robotics and Automation (ICRA), 12, 13.
- **Johannsmeier, L.**, & Haddadin, S. (2015). A framework for task allocation in collaborative industrial assembly processes. In 8th International Workshop on Human-Friendly Robotics.

### Book Chapters

- Aßmann, U., Chen, L., Ebert, S., Göhringer, D., Grzelak, D., Hidalgo, D., ..., **Johannsmeier, L.**, ... & Podlubne, A. (2021). Human-robot cohabitation in industry. In Tactile Internet (pp. 41-73). Academic Press.

### Submission in Preparation

- **Johannsmeier, L.**, Schneider, S., Voigt, F., & Haddadin, S. (2024). Motor Memory for Few-Shot Learning in Robotic Manipulation. In preparation.
- **Johannsmeier, L.**, Schneider, S., Li, Y., Burdet, E., & Haddadin, S. (2024). A Process-Centric Manipulation Taxonomy for the Organisation, Classification and Synthesis of Tactile Robot Skills. Submitted to Nature Machine Intelligence.



# 2

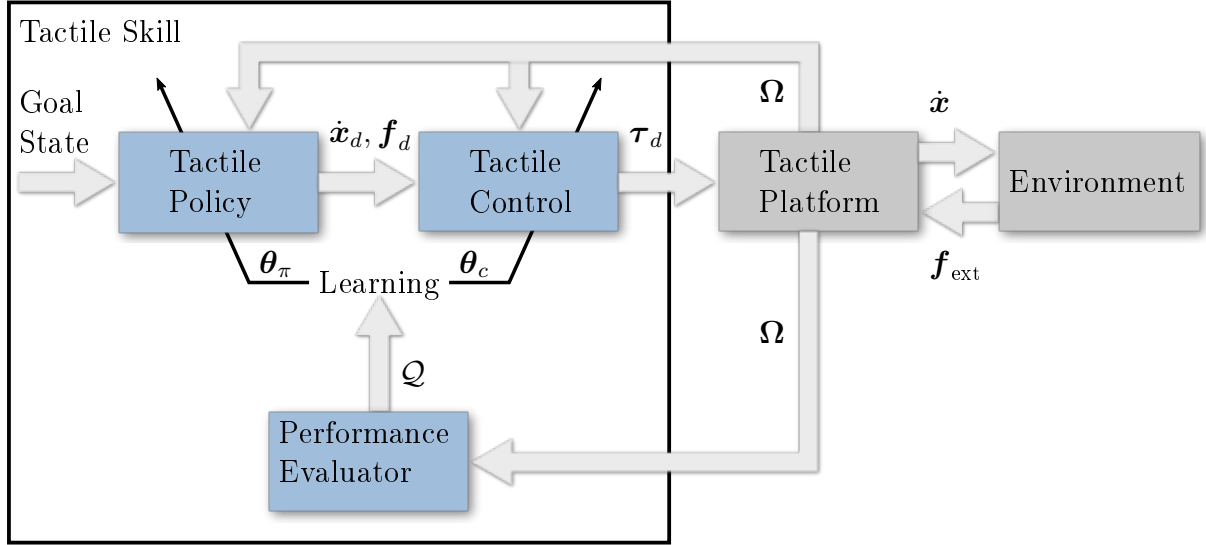
## Theoretical Foundations

This chapter introduces the theoretical foundations on which the proposed learning architecture is built. The fundamental element of the framework is the tactile skill formalism described in Sec. 2.1 consisting of the tactile platform, tactile controller, tactile policy, and a performance evaluator. As a complementary component, a formal process definition is provided that describes industrial processes in terms of manipulation steps and boundary conditions such as error and success states. In order to connect these two elements, a taxonomy is devised with a hierarchical structure that organizes tactile policies according to process properties. Based on the tactile skill, process definition, and taxonomy, a synthesis procedure is presented that automatically selects a tactile policy that is suited to solve a given input process. In Sec. 2.3 an assembly planner is introduced that makes direct use of the tactile skill concept and extends it to skill sequences. The planner solves the allocation problem for a team of humans and robots for a given assembly problem. Finally, in Sec. 2.4 the basis for the learning architecture introduced in Ch. 3 is introduced. It presents a number of state-of-the-art learning algorithms that are used throughout this thesis as well as useful performance metrics to realize the performance evaluator. Finally, a robot motor memory effect is described that was discovered in an earlier experiment. The effect forms the basis for the transfer learning experiments described in Ch. 4. This chapter was written based on [14, 23–25].

### 2.1 Representation

#### 2.1.1 Tactile Skill

Tactility [295] refers to the capability to perceive external stimuli through touch in the form of, for example, force, surface texture, or hardness. In this work, a tactile skill is referred to as a computational complex consisting of policy, control, and learning that, together with a tactile platform, constitutes the system class of tactile robots that is introduced in [296]. Tactile perception systems match corresponding levels in the fundamentally underlying physical event chain from contact pressure distribution to joint



**Figure 2.1:** Tactile Skill Architecture.  $\hat{\mathbf{x}}_d$  is the desired twist,  $\mathbf{f}_d$  is the desired wrench,  $\boldsymbol{\tau}_d$  is the desired joint torques,  $\hat{\mathbf{x}}$  is the actual twist of the robot,  $\mathbf{f}_{\text{ext}}$  is the measured external wrench,  $\boldsymbol{\Omega}$  is the percept vector,  $Q$  is the performance of the skill,  $\boldsymbol{\theta}_\pi$  is the policy parameters, and  $\boldsymbol{\theta}_c$  is the controller parameters.

torque, ranging from joint torque sensors to endpoint force/torque sensors and high fidelity tactile skin feedback. Figure 2.1 depicts the overall architecture of a tactile skill, which is composed of three fundamental components and closely connected to a fourth one as follows.

**I. Tactile Platform** The interaction of a tactile platform (i.e., the robot's hardware) with its environment is determined by the power pair  $\langle \hat{\mathbf{x}}, \mathbf{f}_{\text{ext}} \rangle$ , where  $\hat{\mathbf{x}}$  is the twist of the robot's end effector and  $\mathbf{f}_{\text{ext}}$  is the external wrench. The dynamics of a robot that interacts with the environment is

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_d + \boldsymbol{\tau}_{\text{ext}}, \quad (2.1)$$

where  $\mathbf{M}(\mathbf{q})$  is the mass matrix,  $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$  is the coriolis matrix,  $\mathbf{g}(\mathbf{q})$  is the gravity vector,  $\mathbf{q}$  is the joint position,  $\dot{\mathbf{q}}$  is the joint velocity,  $\ddot{\mathbf{q}}$  is the joint acceleration, and  $\boldsymbol{\tau}_d$  is the desired motor torque.  $\boldsymbol{\tau}_{\text{ext}}$  is the result of the physical event chain from the surface pressure distribution  $\mathbf{u}$ , which originates from a contact, to effective external joint torques in generalized coordinates

$$\boldsymbol{\tau}_{\text{ext}} = \mathbf{f}_1(\mathbf{f}_2(\mathbf{f}_3(\mathbf{f}_4(\mathbf{u})))), \quad (2.2)$$

where  $\{\mathbf{f}_i\}$  denote the subsequent stages of the physical event chain. The percept vector  $\boldsymbol{\Omega}$  encodes the tactile platform's proprioceptive and exteroceptive sensor measurements, including tactile sensation. It is defined as the product of sensor matrix  $\boldsymbol{\Psi}$  and physical state vector  $\boldsymbol{\varsigma}$

$$\boldsymbol{\Omega} := \boldsymbol{\Psi}\boldsymbol{\varsigma}(\mathbf{u}). \quad (2.3)$$

The physical state vector

$$\boldsymbol{\varsigma}(\mathbf{u}) = \begin{bmatrix} \mathbf{u} := \{\{\sigma_j^j, \xi_j^j\}_{j=1\dots k_i}\}_{i=1\dots n} \\ \{\{\mathcal{F}_j^i\}_{j=1\dots k_i}\}_{i=1\dots n} = \mathbf{f}_4(\mathbf{u}) \\ \{\mathcal{F}_i^i\}_{i=1\dots n} = \mathbf{f}_3(\{\{\mathcal{F}_j^i\}_{j=1\dots k_i}\}_{i=1\dots n}) \\ \{m_{z_i}^i\}_{i=1\dots n} = \mathbf{f}_2(\{\mathcal{F}_i^i\}_{i=1\dots n}) \\ \boldsymbol{\tau}_{\text{ext}} = \mathbf{f}_1(\{m_{z_i}^i\}_{i=1\dots n}) \\ \mathbf{q} \\ \dot{\mathbf{q}} \\ \mathbf{x} := \mathbf{f}_{ik}(\mathbf{q}) \\ \dot{\mathbf{x}} \end{bmatrix} \quad (2.4)$$

contains the physical event chain and the robot state.  ${}^j\sigma_j$  denotes the normal contact stresses and  ${}^j\xi_j$  denotes the strains.  ${}^i\mathcal{F}_j$  is the wrench at link  $i$  of the kinematic chain,  ${}^im_{z_i}$  represents the moments that act at joint  $i$ . The forward mapping  $\mathbf{f}_{ik}$  calculates the end-effector pose  $\mathbf{x}$ .

Note that, in this work, the focus lies on a sensor matrix  $\Psi$  with measurements  $\boldsymbol{\tau}_{\text{ext}}$ ,  $\mathbf{q}$ ,  $\dot{\mathbf{q}}$ ,  $\mathbf{x}$ , and  $\dot{\mathbf{x}}$ . This is technologically feasible with state-of-the-art systems. However, future work will also cover  $\Omega$  with extensive sensing abilities of tactile skins, including interaction vibrations, temperature, surface humidity or chemistry, [297, 298], as well as individual Cartesian link state measurements (which were introduced in [50, 299]).

**II. Tactile Controller** A tactile controller - typically a unified force/impedance controller [8, 108] or an adaptive impedance controller [9] - receives  $\Omega$  and computes the desired joint torques  $\boldsymbol{\tau}_d$ .

### Unified Force / Impedance Controller

The most basic form is a parametric force / impedance controller that is able to adapt its stiffness via  $\boldsymbol{\theta}_{c,K}$  with a feedforward wrench and a desired wrench via  $\boldsymbol{\theta}_{c,f}$

$$\begin{aligned} \boldsymbol{\tau}_d(\boldsymbol{\theta}_c) = & \mathbf{J}_x^T(\mathbf{q})[-\mathbf{K}(\boldsymbol{\theta}_{c,K})\mathbf{e}(t) - \mathbf{D}[\mathbf{M}(\mathbf{q}), \mathbf{K}(\boldsymbol{\theta}_{c,K})]\dot{\mathbf{e}}(t) - \mathbf{g}(\mathbf{q}) + \\ & \mathbf{k}_p(\boldsymbol{\theta}_{c,f})(\mathbf{f}_d(t) - \mathbf{f}_{\text{ext}}) + \mathbf{k}_d(\boldsymbol{\theta}_{c,f})(\dot{\mathbf{f}}_d(t) - \dot{\mathbf{f}}_{\text{ext}}) + \mathbf{k}_i(\boldsymbol{\theta}_{c,f})\mathbf{h}_i(\mathbf{f}_{\text{ext}}, t)] \\ & + \mathbf{f}_{ff}(\boldsymbol{\theta}_{c,f}, t), \end{aligned} \quad (2.5)$$

where  $\mathbf{K}(\cdot)$  is the stiffness matrix,  $\mathbf{D}(\cdot)$  is the damping matrix,  $\mathbf{f}_{ff}$  is a feedforward wrench,  $\mathbf{J}_x(\cdot)$  is the Jacobian (defined by  $\dot{\mathbf{x}} = \mathbf{J}_x\dot{\mathbf{q}}$ ),  $\mathbf{e} = \mathbf{x}_d - \mathbf{x}$  is the pose error, and  $\dot{\mathbf{e}}$  is the velocity error.  $k_p(\cdot)$ ,  $k_i(\cdot)$  and  $k_d(\cdot)$  are the force controller parameters, and  $\mathbf{f}_e = \mathbf{f}_d - \mathbf{f}$  is the force error.  $\mathbf{h}_i$  is defined as

$$\mathbf{h}_i(\mathbf{f}_{\text{ext}}, t) = \int_0^t (\mathbf{f}_d(\sigma) - \mathbf{f}_{\text{ext}}(\sigma))d\sigma \quad (2.6)$$

### Adaptive Force / Impedance Controller

Another controller that is used in the experiments in Sec. 4.2.1 is the adaptive force / impedance controller given by

$$\boldsymbol{\tau}_d = \mathbf{J}_x(\mathbf{q})(-\mathbf{f}_{ff}(t) - \mathbf{f}_d(t) - \mathbf{K}_x(t)\mathbf{e} - \mathbf{D}_x[\mathbf{M}(\mathbf{q}), \mathbf{K}(t, \mathbf{K})]\dot{\mathbf{e}}) \quad (2.7)$$

where  $\mathbf{f}_{ff}(t)$  is the adaptive feed forward wrench,  $\mathbf{f}_d(t)$  is an additional predefined type-dependent wrench trajectory that encodes potential prior knowledge,  $\mathbf{K}_x(t)$  the adaptive stiffness matrix,  $\mathbf{D}_x$  the damping matrix, and  $\mathbf{J}_x$  the Jacobian. The position and velocity error are  $\mathbf{e} = \mathbf{x}_d - \mathbf{x}$  and  $\dot{\mathbf{e}} = \dot{\mathbf{x}}_d - \dot{\mathbf{x}}$ , respectively. The adaptive feed forward wrench  $\mathbf{f}_{ff}(t)$  and stiffness  $\mathbf{K}_x$  are

$$\mathbf{f}_{ff}(t) = \int_0^t \dot{\mathbf{f}}_{ff}(t) dt + \mathbf{f}_{ff,0} \quad (2.8)$$

$$\mathbf{K}_x(t) = \int_0^t \dot{\mathbf{K}}_x(t) dt + \mathbf{K}_{x,0} \quad (2.9)$$

where  $\mathbf{f}_{ff,0}$  and  $\mathbf{K}_{x,0}$  denote the initial values. The controller adapts feed forward and stiffness by

$$\dot{\mathbf{f}}_{ff}(t) = \frac{1}{T} \text{diag}(\boldsymbol{\alpha})(\epsilon - \text{diag}(\boldsymbol{\gamma}_\alpha(t))\mathbf{f}_{ff}(t)) \quad (2.10)$$

$$\dot{\mathbf{K}}_x(t) = \frac{1}{T} \text{diag}(\boldsymbol{\beta})(\text{diag}(\epsilon \circ \mathbf{e}) - \text{diag}(\boldsymbol{\gamma}_\beta(t))\mathbf{K}_x(t)) \quad (2.11)$$

The positive-definite meta parameter vectors  $\boldsymbol{\alpha}$ ,  $\boldsymbol{\beta}$ ,  $\boldsymbol{\gamma}_\alpha$ ,  $\boldsymbol{\gamma}_\beta$  are the learning rates for feed forward commands, stiffness and the respective forgetting factors. The learning rates  $\boldsymbol{\alpha}$  and  $\boldsymbol{\beta}$  determine stiffness and feed-forward adaptation speed.  $\boldsymbol{\gamma}_\alpha$  and  $\boldsymbol{\gamma}_\beta$  are responsible for slowing down the adaptation process, which is the main dynamical process for low errors.  $T$  denotes the sample time of the controller.

In order to constrain the subsequent meta learning problem the following well-known constraints are used that characterize essentially every physical real-world system. For better readability the scalar case is discussed, which generalizes trivially to the multi-dimensional one. The first constraint of an adaptive impedance controller is the upper bound of stiffness adaptation speed

$$\dot{K}_{\max} = \max_{t>0} \frac{1}{T} [\beta(\epsilon(t)e(t) - \gamma_\beta K(t))] \quad (2.12)$$

If it is assumed that  $K(t=0) = 0$  and  $\dot{e}$  then  $e_{\max}$  may be defined as the error at which  $\dot{K}_{\max} = \frac{\beta e_{\max}^2}{T}$  holds. Then, the maximum value for  $\beta$  can be written as

$$\beta_{\max} = \frac{\dot{K}_{\max} T}{e_{\max}^2} \quad (2.13)$$

Furthermore, the maximum decrease of stiffness occurs for  $e = 0$  and  $K(t) = K_{\max}$ , where  $K_{\max}$  denotes the maximum stiffness, also being an important constraint for any real-world impedance controlled robot. Thus, an upper bound for  $\gamma_\beta$  may be calculated as

$$\gamma_{\beta,\max} = \frac{\dot{K}_{\max}}{\beta_{\max} K_{\max}} \quad (2.14)$$

Finding the constraints of the adaptive feed forward wrench may be done analogously. In conclusion, the upper limits of the meta parameters  $\boldsymbol{\alpha}$ ,  $\boldsymbol{\beta}$ ,  $\boldsymbol{\gamma}_\alpha$ ,  $\boldsymbol{\gamma}_\beta$  can be related to the inherent system constraints  $\mathbf{K}_{x,\max}$ ,  $\mathbf{f}_{\max}$ ,  $\dot{\mathbf{K}}_{x,\max}$ , and  $\dot{\mathbf{f}}_{\max}$ .



**III. Tactile policy** The tactile controller follows a suitable tactile policy  $\pi_d(\mathbf{\Omega}, \boldsymbol{\theta}_\pi) = [\dot{\mathbf{x}}_d, \dot{\mathbf{f}}_d]$  that uses the percept feedback  $\mathbf{\Omega}$  and the policy parameters  $\boldsymbol{\theta}_\pi$  to learn and generate a global behavior with a local reaction ability, see Fig. 2.1. This tactile policy may be the solution of a virtual impedance system as described in [9], a tactile dynamic movement primitive [300], or a basic ODE system that inherently captures a coordinated motion and wrench reference through the implicit functions  $F_x$  (twist commands) and  $F_f$  (wrench commands), i.e.,

$$F_x[\boldsymbol{\theta}_\pi, \mathbf{\Omega}, \mathbf{x}_d(t), \dot{\mathbf{x}}_d(t), \ddot{\mathbf{x}}_d(t)] = \mathbf{0} \quad (2.15)$$

$$F_f[\boldsymbol{\theta}_\pi, \mathbf{\Omega}, \mathbf{f}_d(t), \dot{\mathbf{f}}_d(t), \ddot{\mathbf{f}}_d(t)] = \mathbf{0}. \quad (2.16)$$

where  $\mathbf{x}_d(t)$ ,  $\dot{\mathbf{x}}_d(t)$ , and  $\ddot{\mathbf{x}}_d(t)$  are the desired pose, velocity, and acceleration, respectively, and  $\mathbf{f}_d(t)$ ,  $\dot{\mathbf{f}}_d(t)$ , and  $\ddot{\mathbf{f}}_d(t)$  are the desired wrench and its derivatives.

**IV. Performance Evaluator** Finally, the performance evaluator measures the skill performance  $\mathcal{Q}$  based on a set of  $n$  performance measures  $\{\mathcal{Q}_i(\mathbf{\Omega})\}$  weighted with  $\{\omega_i\}$ .

$$\mathcal{Q}(\mathbf{\Omega}) = \sum_{i=1}^n \mathcal{Q}_i(\mathbf{\Omega}) \omega_i, \quad \text{where} \quad \sum_{i=1}^n \omega_i = 1. \quad (2.17)$$

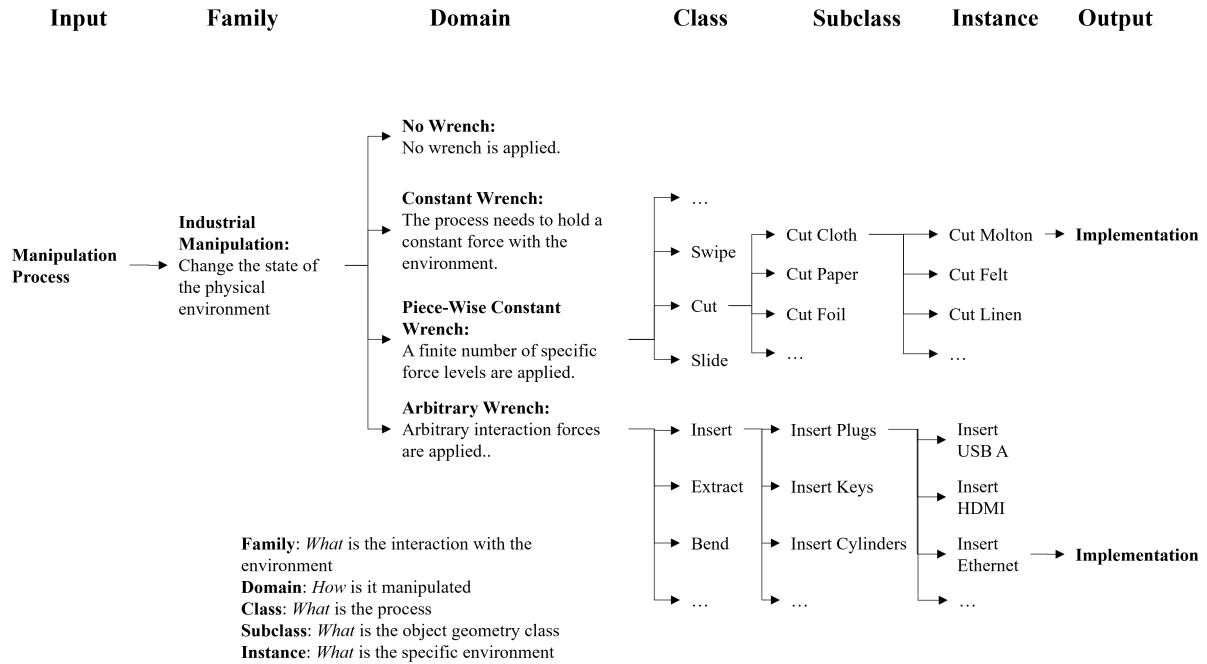
### 2.1.2 Process Definition

A process  $p$  is defined by the steps that are needed to achieve the process objective and by its boundary conditions. There are four process states; namely, the initial state  $s_0$ , the error state  $s_e$ , the final state  $s_1$ , and the policy state  $s_\pi$ . The policy state  $s_\pi$  contains a number of substates, which are connected by transitions  $\delta \in \Delta$ , where  $\Delta$  is the set of all transitions in  $p$ . The substates are later implemented by a compatible skill framework. Three boundary conditions determine the switch between the top-level states, and the transitions determine the switch between the policy states:

- The precondition  $\mathcal{C}_{\text{pre}}(\mathcal{O}) = c_{1,\text{pre}}(\mathcal{O}) \wedge \dots \wedge c_{n_{\text{pre}},\text{pre}}(\mathcal{O})$  checks whether the process is ready to start, and it switches from  $s_0$  to  $s_\pi$ .
- The error condition  $\mathcal{C}_{\text{err}}(\mathcal{O}) = c_{1,\text{err}}(\mathcal{O}) \vee \dots \vee c_{n_{\text{err}},\text{err}}(\mathcal{O})$  is triggered in the case of irreversible failure, and it immediately terminates the process, leading to the error state  $s_e$ . It switches from  $s_0$  or  $s_\pi$  to  $s_e$ .
- The success condition  $\mathcal{C}_{\text{suc}}(\mathcal{O}) = c_{1,\text{suc}}(\mathcal{O}) \wedge \dots \wedge c_{n_{\text{suc}},\text{suc}}(\mathcal{O})$  indicates the success of the process. It switches from  $s_0$  or  $s_\pi$  to  $s_1$ .
- A substate  $s_{\pi_i}$  switches to another substate  $s_{\pi_j}$  through a connecting transition  $\delta$  if a number of conditions is fulfilled, i.e. each transition is an expression  $\delta(\mathcal{O}) = \delta_1(\mathcal{O}) \wedge \dots \wedge \delta_{n_\delta}(\mathcal{O})$ .

$n_{\text{pre}}$ ,  $n_{\text{err}}$ ,  $n_{\text{suc}}$ , and  $n_\delta$  indicate the respective number of conditions. The boundary conditions and transitions map logical conjunctions to the interval  $[0, 1]$  where 0 means false and 1 means true. In case of a true evaluation a state switch is triggered. The conditions depend on a set of objects  $\mathcal{O}$  that form the environment for the process. An object  $o \in \mathcal{O}$  is characterized by at least its Cartesian pose  $T_o$ , and possibly other physical properties such as mass, center of mass, inertia, etc. All objects have a unique identifier (e.g., *Object*) and a handle (e.g.,  $o_1$ ).

### 2.1.3 Taxonomy Structure



**Figure 2.2:** Ranks of the taxonomy of manipulation skills

A constructive end-to-end manipulation framework with three distinct structural elements is introduced. The *manipulation process definition* constitutes the abstraction and the interface that is compatible with the *taxonomy of manipulation skills* (TMS), encoding the skill selection process. The actual implementation is realized by a *process-compatible tactile skill framework*, which can utilize machine learning while still maintaining stability and guaranteeing safety in its control stack. With the help of this three-element approach, it is possible for a user or an autonomous planning system to select reliable tactile skills that perform well in industrial contexts. The skills can be used and optimized without domain-specific robotics or machine learning knowledge.

The process definition is the input to the taxonomy. It is the interface for process experts such as technicians, robot operators, or shop-floor workers to frame their process knowledge. Overall, this abstraction is inherently compatible with AI planning systems such as the PDDL-based FastDownward planner [301], or the ASP-based clingo [302].

The developed TMS connects the two domains of process-centric and robot-centric representations. Specifically, it allows us to map a given process definition to a unique tactile skill model using its underlying classification scheme. The TMS contains five taxonomic ranks (see Fig. 2.2) in total; namely, *family*, *domain*, *class*, *subclass*, and *instance*. Each rank represents a decision layer that reduces the set of possible solution policies for a given input process until a complete solution is determined. The family rank determines the type of interaction with the environment. So far, only processes with force interaction were considered in the taxonomy. The domain rank separates policies on the basis of the interaction forces between the process objects and the environment. At the class rank, further refinement is done based on well-defined, concrete manipulation steps that lead from the initial state of the process to its final state. The subclass rank groups the skill solutions according to the distinct geometries of the manipulated objects (e.g., plugs and

keys). Finally, the instance rank contains skill solutions that represent an input process with a concrete physical setup. Note though that the configuration of the setup (i.e. the object poses) is not defined, meaning that one skill instance can have arbitrarily many tasks, i.e. physical instantiations.

Input processes for the TMS are directly derived from established standards such as the German curricula for trainees in metalworking [303], electronics [304], and mechatronics [305]. These standards provide the basis for almost any process in today's industry by defining boundary conditions, manipulation steps, requirements, and objectives. By building on top of standard works, the presented framework is directly compatible with the current needs of industrial companies. As a first step, the TMS contains processes that range from machine tending (e.g. operating levers and pressing buttons), to assembly (e.g. insertion), or material processing (e.g. bending and cutting).

## 2.2 Tactile Skill Synthesis Procedure

A synthesis procedure was devised to formally close the gap between process definition and skill implementation. The selection of a robot manipulation policy  $\pi_d(\Omega, \theta_\pi)$  from a desired manipulation process  $p \in \mathcal{P}$  is expressed by

$$\mathcal{T} = \mathcal{T}_1 \circ \mathcal{T}_2 \circ \mathcal{T}_3 \circ \mathcal{T}_4 : p \rightarrow \pi_d, \quad (2.18)$$

where  $\mathcal{T}$  is the taxonomic algorithm that maps  $p$  to a unique  $\pi_d(\cdot)$ .  $\pi_d$  is then learned with a suitable algorithm (i.e., the policy and controller parameters  $(\theta_\pi, \theta_c)$  are learned) so that  $\pi_d^* = \pi_d(\Omega, \theta_\pi^*)$  and  $\tau_d^* = \tau_d(\theta_c^*)$  are the optimal policy and controller, solving  $p$ . The algorithm steps  $\mathcal{T}_i$  correspond to the ranks in the taxonomy<sup>1</sup>.

$\Pi_0$  is the initial set of all available policies.  $\mathcal{T}$  iteratively narrows down  $\Pi_0$  to a single  $\pi_d$  by the following steps, which are currently being done manually but could be automated:

1. The *Domain* rank ( $\mathcal{T}_1$ ) selects policies based on their desired wrench  $\mathbf{f}_d$ :  
 $\Pi_{1,1} = \{\pi_d \mid \mathbf{f}_d(t) = \mathbf{0} \forall t\}$ ,  $\Pi_{1,2} = \{\pi_d \mid \mathbf{f}_d(t) = \text{Const.} \forall t\}$ ,  
 $\Pi_{1,3} = \{\pi_d \mid \mathbf{f}_d(t) \in \{\mathbf{f}_{d,1}, \dots, \mathbf{f}_{d,n}\} \forall t\}$ ,  $\Pi_{1,4} = \{\pi_d \mid \mathbf{f}_d(t) \in \mathbb{R} \forall t\}$
2. The *Class* rank ( $\mathcal{T}_2$ ) selects policies that reach  $s_1$ :  
 $\Pi_{1,i,j} = \{\pi_d \mid w(t) = s_1 \text{ for } t \rightarrow \infty\}$
3. The *Subclass* rank ( $\mathcal{T}_3$ ) selects policies that follow the process steps defined by  $\Delta$ :  
 $\Pi_{i,j,k} = \{\pi_d \mid \delta \rightarrow \text{true} \forall \delta \in \Delta\}$
4. The *Instance* rank ( $\mathcal{T}_4$ ) selects the policy with the lowest number of parameters:  
 $\pi_d = c(\{\pi_d \mid \min_{|\theta_\pi|} \pi_d \in \Pi_{i,j}\})$ .  
 $c$  represents a choice if more than one policy is left. Furthermore, the parameter domain  $\mathbb{D}$  is determined using constraints from the system and the process instance:  
 $\mathbb{D} = f_{\mathbb{D}}(\theta_\pi, \theta_c, \mathbb{C})$   
with constraints  
 $\mathbb{C} = \{\tau_{\text{ext,max}}, \mathbf{f}_{\text{ext,max}}, t_{\text{max}}, \dot{\mathbf{x}}_{\text{max}}, \ddot{\mathbf{x}}_{\text{max}}\}$   
This (so far manual) step is represented by  $f_{\mathbb{D}}$ , i.e.

<sup>1</sup>Note that the family rank is omitted since, so far, only one element exists on its level.

For two example processes, inserting an Ethernet plug and cutting a piece of cloth, the synthesis procedure is described below. More details such as the available set of policies  $\Pi$  can be found in App. A.1. Note that in the following  $\mathcal{U}(T)$  means an environment around a Cartesian pose  $\mathbf{T}$  that is defined by intervals in  $x$ ,  $y$ , and  $z$  as well as in  $\text{SE}(3)$ . Furthermore, boundary conditions and transitions are expressed as equations that map a number of conditions connected by conjunctions to  $[0, 1]$  (false or true).

### Synthesis: Inserting an Ethernet Plug

Process definition:

$$\mathcal{O} = \{o_1, o_2, o_3\}, \mathcal{C}_{\text{pre}} = \emptyset, \mathcal{C}_{\text{err}} = \emptyset, \mathcal{C}_{\text{suc}} = T_{o_1} \in \mathcal{U}(o_3),$$

$$\Delta = \{\delta_{1,2} = T_{o_1} \in \mathcal{U}(o_2), \delta_{2,3} = \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{contact}}\}$$

1. *Domain* ( $\mathcal{T}_1$ ): The process involves a search process with complex interaction forces. Therefore  $\Pi_{1,4}$  is selected:

$$\Pi_{1,4} = \{\pi_{d,13}, \pi_{d,14}, \pi_{d,15}, \pi_{d,16}, \pi_{d,17}, \pi_{d,18}, \pi_{d,19}, \pi_{d,20}, \pi_{d,27}, \pi_{d,28}, \pi_{d,32}\}$$

2. *Class* ( $\mathcal{T}_2$ ): The policies that can reach the final state  $s_1$  of the insert process are

$$\Pi_{1,4,1} = \{\pi_{d,17}, \pi_{d,20}, \pi_{d,27}, \pi_{d,28}, \pi_{d,32}\}$$

3. *Subclass* ( $\mathcal{T}_3$ ): The policies that are not guaranteed to reach the process's substates are removed:

$$\Pi_{1,4,1} = \{\pi_{d,27}, \pi_{d,28}, \pi_{d,32}\}$$

4. *Instance* ( $\mathcal{T}_4$ ): From  $\Pi_{1,4,1}$ , the least complex policy is selected to be  $\pi_{d,27}$ .

### Synthesis: Cutting Cloth

Process definition:

$$\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5\}, \mathcal{C}_{\text{pre}} = \emptyset, \mathcal{C}_{\text{err}} = \{\mathbf{f}_{\text{ext},z} < \mathbf{f}_{\text{cut}}\}, \mathcal{C}_{\text{suc}} = T_{o_1} \in \mathcal{U}(o_5),$$

$$\Delta = \{\delta_{1,2} = T_{o_1} \in \mathcal{U}(o_2), \delta_{2,3} = \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{cut}}, \delta_{3,4} = T_{o_1} \in \mathcal{U}(o_4)\}$$

$\mathbf{f}_{\text{cut}}$  is the desired cutting force.

1. *Domain* ( $\mathcal{T}_1$ ): The process requires a constant cutting force, but it also has phases without any contact. Therefore,  $\Pi_{1,3}$  is selected:

$$\Pi_{1,3} = \{\pi_{d,22}, \pi_{d,24}, \pi_{d,26}, \pi_{d,30}, \pi_{d,31}\}$$

2. *Class* ( $\mathcal{T}_2$ ): The policies that can reach the final state  $s_1$  of the cut process are

$$\Pi_{1,3,2} = \{\pi_{d,26}, \pi_{d,31}\}$$

3. *Subclass* ( $\mathcal{T}_3$ ): The policies that are not guaranteed to reach the process's substates are removed:

$$\Pi_{1,3,2} = \{\pi_{d,26}\}$$

4. *Instance* ( $\mathcal{T}_4$ ): From  $\Pi_{1,3,2}$ , the least complex policy is selected to be  $\pi_{d,26}$ .

## 2.3 Artificial Intelligence-Based Assembly Planning

This section describes the representation of assembly plans and a search method to solve them.

### 2.3.1 Introduction

In the following, industrial assembly processes are considered that are typically well understood and planned in advance. Therefore, the construction plans are known beforehand. Also, it is assumed that during task execution all necessary tools and parts are fed to the team by suitable mechanisms such as conveyor belts. In turn, the requirements for autonomy are restricted to specific problems such as being able to execute a set of assembly and manipulation skills, or to communicate with humans and other robots within the context of assembly processes.

A framework for human-robot collaboration is proposed that comprises three different architectural levels: the team-level assembly task planner, the agent-level skill planning, and the skill execution level that is the final decision component above the robot real-time control stack.

The planner at team level performs the task allocation for the agents based on an abstract world model and with the help of suitable cost metrics. To suitably model the types of assembly processes AND/OR graphs [114] are employed as they implicitly model parallelism. The team-level planner produces task sequences for every agent via A\* graph-search, from which it derives task descriptions that are then passed down to the agent's skill execution level. The agents in turn implement tactile skills as introduced in Sec. 2.1. The proposed separation in terms of entities (agents, team-level planner) and abstraction leads to an efficient planning framework that can produce optimal nominal task plans to build a desired assembly, while handling failures due to dynamic and uncertain environments in a safe and reliable way on the lower levels of control.

#### 2.3.1.1 Preliminaries

To be able to create (large) assembly plans and design control processes that implement those plans, it is necessary to select a proper representation. In this chapter such a representation is briefly introduced and reasonable assumptions are made about the structure of the assembly, simplifying the subsequent theoretical analysis. Note that in this work the generation process of assembly plans is not treated. Instead, it is assumed that such plans are already at disposal. A popular algorithm to generate assembly plans is e.g. described in [306].

In the following, a mechanical assembly is denoted by  $\mathcal{A}$ . An intuitive representation for a single assembly is the graph of connections  $G$ . This is obtained by simplifying the graph of contact, which was already used in several works [306–308]. The resulting graph of connection contains a node for every part  $\rho \in \Gamma$ , where  $\Gamma$  is the set of all parts, and an edge if and only if there is at least one contact between the two respective parts.

The parts  $\rho$  are assumed to be solid rigid objects, i.e. their shape remains the same throughout the complete assembly process. A subassembly  $\Gamma_s \subset \Gamma$  has similar properties as the complete assembly. If a subassembly consists of more than one part, all the parts are connected, meaning the graph of connections induced by that subassembly is connected.

For the sake of simplicity it is assumed that every part  $\rho$  is unique, i.e. they are not interchangeable when they have equal properties (e.g. two screws of the same type). Furthermore, it is assumed that the geometry and characteristics of the connection of every pair of parts are unique. Thus, every subassembly implicitly defines the complete geometry of the involved parts as well as their connections. A subassembly with only one part is considered atomic.

In order to build an assembly an assembly plan is needed, which describes the possibilities of how to assemble a work piece. In particular, it assumes that all parts are in their designated places and the necessary connections were made already, so that one begins with the correct subassemblies. The process of executing the assembly plan is called assembly process. The process of assigning the available workers  $w \in W$  (for which also the more general term agent is used) to the assembly actions is called task allocation. A sequence of assembly actions  $a$  that lead from the initial configuration to the finished product is called an assembly sequence  $\alpha$ . An assembly action  $a$  denotes one step in the assembly process and can be understood as a skill instantiation.

### 2.3.1.2 Assumptions

Three predicates are introduced to further concretize the scenario which is being investigated. First, it is assumed that a subassembly  $\Gamma_s$  is stable,  $st(\Gamma_s)$ , i.e. its parts remain in contact without applying any external forces and the relative geometry of all parts does not change. Then, the actions  $a$  that are used in the assembly process are classified as geometrically feasible and mechanically feasible:

- $gf(a)$  denotes that there exists a collision free path to join the two involved subassemblies.
- $mf(a)$  holds if it is possible to permanently establish all necessary connections.

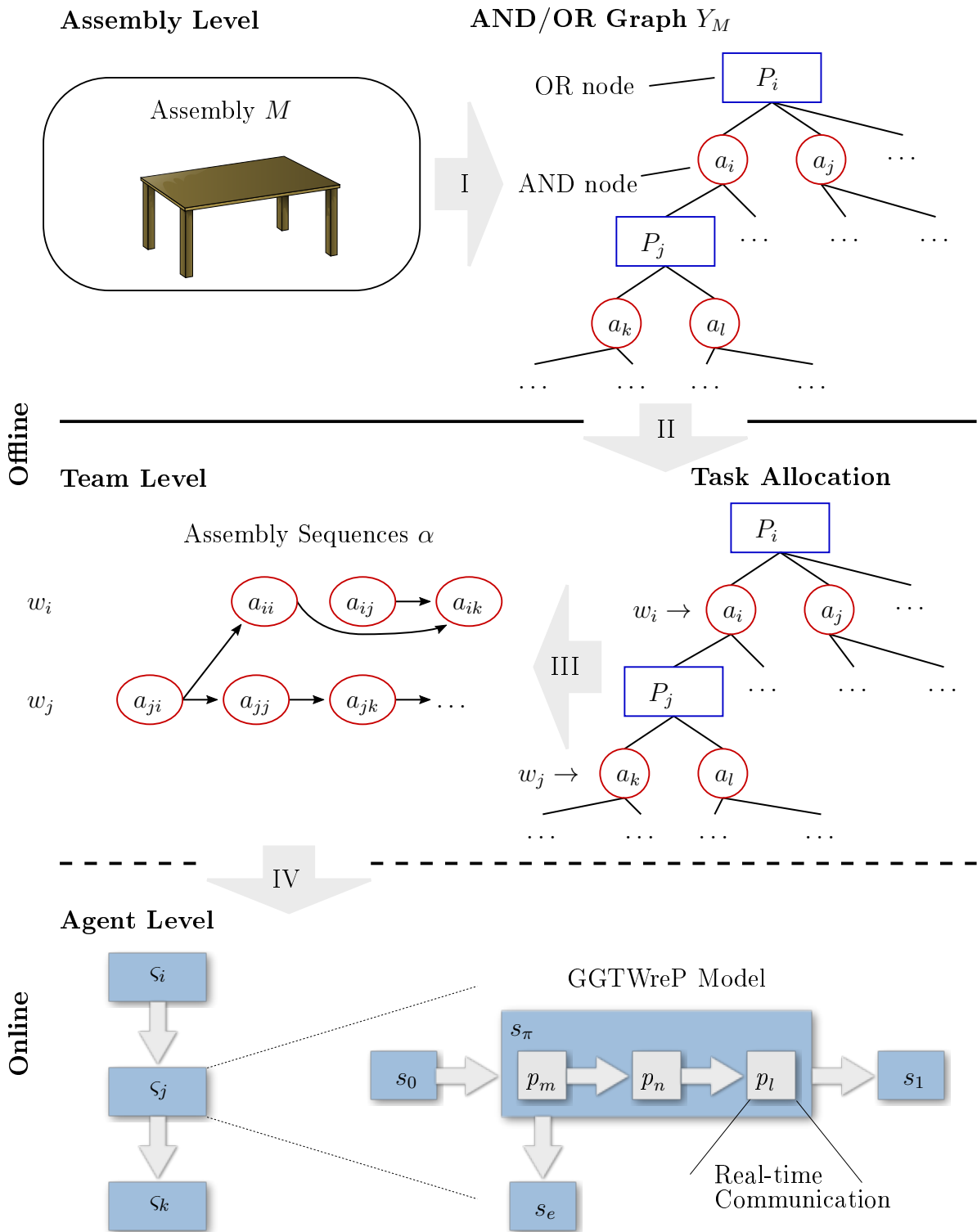
**Sequentiality** A plan is sequential if there exists a sequence of assembly actions, involving only one subassembly that can represent the plan. This means that it is never necessary, though possible, to move two subassemblies at a time. A plan that holds this property could thus be executed by a robot with a single manipulator. Whether an assembly can be described by a sequential plan depends on its geometry. Also, it requires that every subassembly is stable.

**Monotonicity** A plan is called monotone if it contains no action that would break an already established connection or modifies the relative geometry in an already existing subassembly. Monotonicity is often built into real world assemblies.

**Coherency** A plan is coherent for a graph  $G$  if every subassembly occurring in the plan corresponds to a connected sub graph of  $G$ . More formally, there exists a connected graph  $G$  such that

- there exists an isomorphism from the set of parts to the set of nodes in  $G$  and
- for every subassembly occurring in the plan, the sub graph of  $G$  that is induced by that subassembly is connected.

2.3.1.3 Collaborative Assembly Planning Framework



**Figure 2.3:** Collaborative assembly planning framework. The top layer depicts the input to the team-level planner, which is an AND/OR graph  $Y_A$  resulting from an assembly  $\mathcal{A}$  (Step I). The planner then solves the problem of optimal task allocation (Step II) for multiple agents considering a given cost function and constructs a set of assembly sequences (Step III). The actions from the sequences are then passed to the respective agents (Step IV), which possess corresponding skills  $\varsigma$ . Skills in turn consist of more basic structures, i.e. atomic actions, which eventually map to the real-time-level.

Figure 2.3 provides an overview of the collaborative assembly planning framework. The used notation  $w \rightarrow a$  denotes a specific allocation of agent  $w$  to action  $a$ . The framework comprises three main levels, which are strictly separated. However, they communicate bilaterally with each other and share common information. Although the approach is considered as rather general in principle, the focus here lies on its application to industrial assembly processes.

The first level, called team level, plans the assembly process from the view of a foreman, i.e. it has information about the possible construction plans, assembly parts, the available agents and other resources. It knows only actions that can manipulate this world model in an abstract way, i.e. it is domain-specific. The planning process on team-level solves the problem of task allocation in the assembly process for a team of human and robotic workers. It should be noted that at this level of abstraction no explicit distinction between human and robotic workers is made, since the team-level planner relays only abstract task descriptions without the need to take into account specific implementations of the necessary skills. It is possible to distinguish between agents by encoding their respective capabilities into cost functions that are used for planning. At this point it is important to state how the planner interacts with the agents. During the search process the planner sends the request to perform a specific action to an agent. The agent answers with the cost and possibly a further request for an interaction, which in turn is integrated into the planning process. If an agent is not able to perform a specific action it returns infinite costs. The planner then uses the received costs from all available agents to determine the optimal task allocation. Please note further, that the planning process at team-level is offline, see Fig. 2.3, hence, fast replanning and refinement methods are out of the planner's scope, despite it might be possible for a manageable assembly complexity.

The second level, called agent level, maps the team-level planning process to the capabilities (skills) of the robot. In general, the agent level may of course contain also other sophisticated planning systems that plan out sequences of skills to accomplish a given objective. However, for executing industrially relevant skills that need a high level of expert knowledge, practical experience shows that their automatic planning instead of semi-handcrafting in collaboration with process experts is the significantly less capable approach as of today. Note that since the agent level is implemented on the robot itself, it is possible to use different robot types. For this, the systems have to provide the necessary formal semantics to the team level, which obviously requires the systems to be able to execute the same actions in principle. Another important task of the agent level is the consistent handling of events such as collisions or human induced interruptions of the assembly process. Although the reaction to the actual event itself is assumed to be handled by the skill level, the agent level either has to deal with the consequences in a way that is consistent with the overall plan on team level, or report a failure in plan execution and request replanning from the team-level planner.

The third level, the skill level, is directly responsible for executing trajectory planning, controllers, etc. For the experimental analysis of the planning system it is realized by the GGTWreP framework (see Sec. 3.2.5).

### 2.3.2 Assembly Plan Representation

In a collaborative scenario where actions are distributed among several agents  $w \in W$ , it may be desirable and more efficient that some of the actions are executed in parallel. In



particular, when humans and robots work together, scenarios become possible where joining subassemblies may require rather complex and delicate procedures that are extremely difficult to automate, while assembly of subassemblies may be very easy to automate. Consequently, one possibility is that the human handles the complex task of joining the subassemblies, while the robotic co-workers prepare the mentioned subassemblies and 'feed' them to the human when needed.

As mentioned above, AND/OR graphs are chosen as the representation of assembly plans because of their ability to explicitly facilitate parallel execution of assembly actions, as well as the time independence of parallel executable actions. Using AND/OR graphs for the representation of assembly sequences was first proposed in [114]. More recent applications can be found in [309]. The AND/OR graph of a particular assembly can be constructed by disassembling the complete assembly until only atomic parts  $\rho$  are left. Having this idea in mind, the graph of feasible assembly actions is built.

**Definition 1: AND/OR graph of feasible assembly sequences** A mechanical assembly is defined by  $\mathcal{A} = (\Gamma, st, gf, mf)$ . The AND/OR graph  $Y_{\mathcal{A}}$  of feasible assembly sequences of assembly  $\mathcal{A}$  is defined as the hypergraph  $(V, E)$ , where

$$\begin{aligned} V &= \{\Gamma_s | \Gamma_s \subseteq \Gamma \wedge st(\Gamma_s)\}, \\ E &= \{\{\Gamma_{s,k}, \Gamma_{s,i}, \Gamma_{s,j}\} | \Gamma_{s,k}, \Gamma_{s,i}, \Gamma_{s,j} \in V\}, \\ &\text{and} \\ \Gamma_{s,k} &= (\Gamma_{s,i} \cup \Gamma_{s,j}) \text{ with } gf(\{\Gamma_{s,i}, \Gamma_{s,j}\}) \wedge mf(\{\Gamma_{s,i}, \Gamma_{s,j}\}). \end{aligned}$$

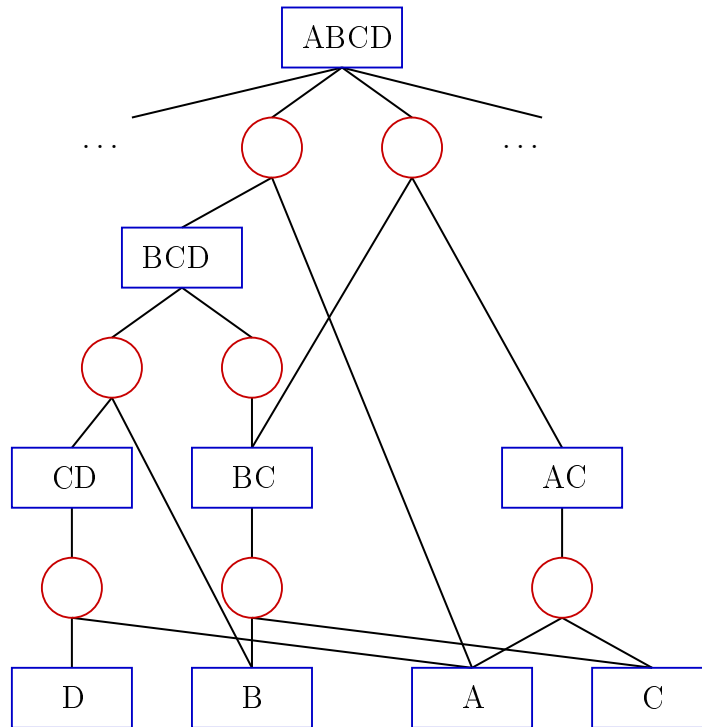
Note that, although each edge in the hypergraph is an unordered set, one of the three subassemblies, namely  $\Gamma_{s,k}$  is distinguished because it is the union of the other two sets  $\Gamma_{s,i}$  and  $\Gamma_{s,j}$ . Figure 2.4 shows an excerpt from an AND/OR graph of a four part assembly consisting of the parts  $\Gamma = \{A, B, C, D\}$ .

### 2.3.3 Task Allocation and Planning

#### 2.3.3.1 Team-Level Task Allocation

The team-level planner takes an assembly plan in the form of an AND/OR graph as input. Furthermore, a reference to a database is needed, which contains the necessary information about part locations, agents and other resources. To optimally solve the allocation problem with respect to a particular cost metric, the well-known A\* graph search algorithm [310] is applied. In the following, the formal problem statement, relevant cost metrics and heuristics, as well as further considerations concerning multi-agent allocation are provided.

**1) Problem Statement** The overall goal is to find an allocation of agents to assembly actions that is optimal with respect to a specified cost function. In the following,  $\mathcal{Z} = \{\zeta_1, \dots, \zeta_n\}$  denotes the set of assembly states, each of which corresponds to a set of OR nodes  $v_o(\zeta) \subset V_o$ , where  $V_o$  denotes the set of all OR nodes in the AND/OR graph  $Y_{\mathcal{A}}$ . Similarly, the assembly actions  $a(\zeta)$  that are applicable in a state  $\zeta$  correspond to a set of AND nodes  $v_a(\zeta) \subset V_a$ , where  $V_a$  denotes the set of all AND nodes in  $Y_{\mathcal{A}}$ . Since  $Y_{\mathcal{A}}$



**Figure 2.4:** Partial AND/OR graph of an exemplary assembly. The blue colored rectangles depict OR nodes, the red colored circles AND nodes.

and the number of available agents, and therefore, the number of possible allocations, are finite, the search space is finite as well.

The general idea is now to propagate through  $Y_{\mathcal{A}}$  from the root node via disassembly actions until some state contains only atomic subassemblies. The following formal definition of the problem complies to the simplifications for the assembly process introduced in Sec. 2.3.1.2.

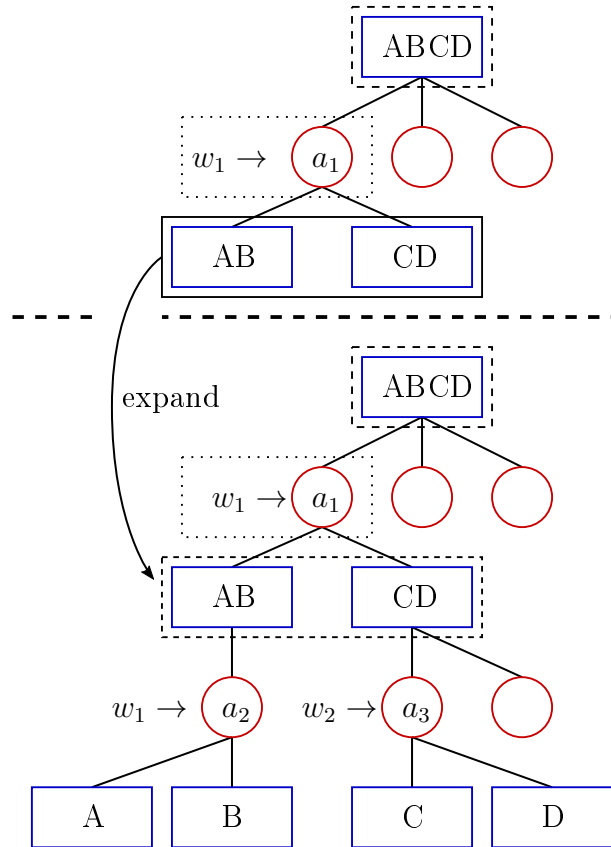
- The initial assembly state  $\zeta_0$  corresponds to the root node  $v_{o,0}$  of  $Y_{\mathcal{A}}$ , which corresponds to the complete assembly  $\mathcal{A}$ .
- In a state  $\zeta$  the number of applicable assembly actions is at most as large as the number of OR nodes  $n_{\zeta,o}$  in that state due to strict sequentiality. Also, a single OR node can provide at most one AND node to a valid set of actions, since a given assembly can not be disassembled into two different configurations at the same time. Therefore, in  $\zeta$  there are  $n_{\zeta,a} = \prod_i n_{a,i}$  different valid sets of actions, where  $n_{a,i}$  denotes the number of children of the  $i$ -th OR node. Also note that an agent  $w$  does not have to be assigned to an action. In addition, every action from  $a(\zeta)$  can be assigned to every agent  $w \in W$ . In a state  $\zeta$ , there is a maximum number of assignments

$$N_A = \sum_{i=1}^l n_{\zeta,a} \binom{n_w}{i} n_{\zeta,o}!, \quad (2.19)$$

where  $l = \min(n_w, n_{\zeta,o})$  is either the number of available workers  $n_w$  or the number of OR nodes with children  $n_{\zeta,o}$ , i.e. whichever is lower.

- The goal state  $\zeta_g$  contains only OR nodes that correspond to atomic subassemblies.
- The cost of an action can be chosen from different possibilities and will be further elaborated below.

To illustrate the propagation of the search algorithm through the search space (i.e. the AND/OR graph) Fig. 2.5 depicts an example.



**Figure 2.5:** Example propagation via search through search space. In the top graph agent  $w_1$  has been assigned to the indicated AND node in state  $\zeta$  and thus, the corresponding assembly action. This AND node has two children that are expanded and form the new state  $\zeta'$ . In the bottom graph two parallel actions are assigned, which again yield two children, respectively. The dashed boxes depict the currently active state, the dotted ones the chosen allocation.

**2) Multi-Agent Considerations** Up to now, no particular assumptions regarding the number of workers were made in the problem statement. Although the AND/OR graph implicitly models parallelism, it is at no point required to have more than one worker available if the simplifications from Sec. 2.3.1.2 are met. Yet, to potentially increase efficiency/speed of the assembly process, a team of agents is used. For this, some factors have to be considered, which are introduced into the planning process at team level.

**a) Actions and Interactions** Within a multi-agent scenario in the domain of collaborative assembly planning, actions are not just used to build the assembly but also to enable interactions between agents. Therefore, it is distinguished between actions that directly modify the assembly and actions that characterize an interaction between two

agents. Also note, that at team level actions are considered to be atomic, i.e. they directly change the model of the world. At agent level the same actions are called skills which in turn map to the skill level. The actions at team level are formally defined as follows.

**Definition 2: Assembly action** Let  $a(\text{type}, W)$  be a general action in the context of assembly processes. Let  $\text{type}$  denote a descriptor that specifies the action type that is requested and  $W$  can either be a single assignment  $W = w_i$ , or a pair assignment  $W = (w_i, w_j)$  depending on the type. Note that interactions, i.e., actions with a pair assignment, with at most two agents are considered.

If e.g.  $\text{type}=\text{hand\_over}$  and  $W = (w_1, w_2)$ , a needed part is not reachable by  $w_1$  and must be provided by  $w_2$  via a hand-over action.

Note that there is no need to change the problem statement to include interactions, it is only necessary to integrate them into the expanded AND/OR graph  $Y_{\mathcal{A}, \zeta}$  that is defined as follows.

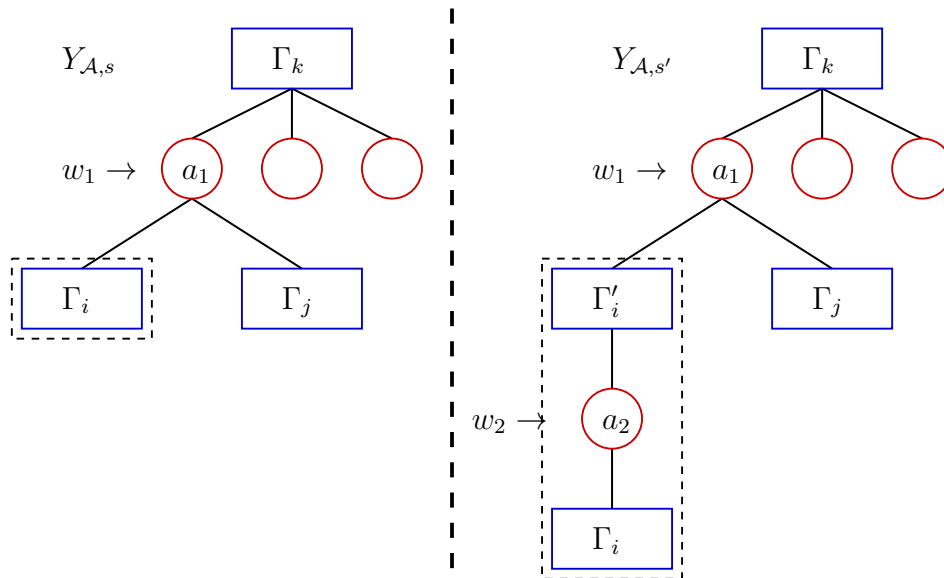
**Definition 3: Expanded AND/OR graph** Let  $Y_{\mathcal{A}, \zeta'}$  be called the local AND/OR graph of the state  $\zeta'$ , then  $Y_{M, \zeta'}$  is created as a copy of  $Y_{\mathcal{A}, \zeta}$  of state  $\zeta$ , which is the predecessor of  $\zeta'$ . The initial  $Y_{\mathcal{A}}$  is the AND/OR graph of  $\zeta_0$ .

Since every interaction is specific to the assigned agent pair and arises from a particular context (e.g. one agent cannot reach a part, yet another can, or one agent needs the assistance of another agent to join two subassemblies), it makes sense to change the AND/OR graph only locally in the search problem, i.e. for the respective state  $\zeta$ . Over a sequence of states  $\zeta$  the graph is expanding with the occurrence of interactions.

If an interaction is necessary, a new OR node with the corresponding subassembly  $\Gamma'_s$  and a new AND node  $v_a$ , which represents the interaction, are inserted, see Fig. 2.6. The new AND node (i.e. the action) can then be assigned to an agent as described above. The new (intermediate) subassembly  $\Gamma'_s$  has the same parts as  $\Gamma_s$  i.e.  $\rho(\Gamma'_s) = \rho(\Gamma_s)$ . However, the state of  $\rho(\Gamma'_s)$  is different from that of  $\rho(\Gamma_s)$  depending on the type of interaction.

**b) Synchronization** Another important aspect in a multi-agent scenario is synchronization. In an ideal situation, the nominal plan would be executed in the real world without any modifications. In practice, this is highly unlikely, in particular due to dynamically changing environments and uncertainty. Thus, it is not possible to guarantee e.g. a successful hand-over action without communication. Since the action sequences arise from the AND/OR graph, every need for synchronization can be dealt with by interactions.

This means that agents do not need to synchronize their actions, as long as their action sequences  $\alpha$  are not connected via an interaction. Since the team-level planner knows (via confirmation from the agents) which actions have been performed, it can let agents wait until the requirements for their next scheduled task are fulfilled. This way only tasks that are applicable are performed. The following example refers to Fig. 4.17, which is also the graph for the experiments in Sec. 4.4. Consider an agent  $w_1$  that performs action  $a_{11}$  and is then scheduled to perform action  $a_7$ . Also consider agent  $w_2$  that performs action  $a_{10}$ .  $w_1$  can not perform  $a_7$  as long  $w_2$  does not confirm the completion of  $a_{10}$ . Hence, the structure of the AND/OR graph represents the agents actions.



**Figure 2.6:** To model an interaction between two agents in the AND/OR graph a new AND/OR node pair is inserted. In this example, the subassembly  $\Gamma_k$  is created by agent  $w_1$  via joining  $\Gamma_i$  and  $\Gamma_j$ , which corresponds to the action  $a_1 := a(\text{assemble}, w_a)$ . Yet, some kind of interaction is necessary to complete the assembly step. E.g., the performing agent cannot reach  $P_i$  so another agent  $w_2$  needs to hand this part over. Therefore,  $\Gamma'_i$  and the AND node that corresponds to the interaction  $a_2 := a(\text{hand\_over}, (w_1, w_2))$  are inserted on the right side.

**3) Optimization Metric and Heuristic** Although the most obvious way to evaluate the cost of an allocation is to measure the overall execution time, there may be other metrics that are more important in specific situations. Furthermore, it is unlikely that time constraints assumed by the team-level planner are adhered to by the human co-workers, since the daily work routine of a factory worker is undetermined over short periods of time. Considering this and the fact that it is not explicitly distinguished between humans and robots at team level, other cost metrics are needed that are more applicable in human-robot collaboration and encode all necessary information about the specific workers. For example, the assembly could involve the handling of dangerous and/or heavy material and thus the human workload and risk should be minimized. Therefore some example cost functions are introduced that are suitable candidates to the given problem:

- Execution Time: One may distinguish between overall execution time and local execution time, i.e. the time needed for the execution of a single action.
- Resource costs: Resources such as energy consumption, peak power, etc. could be taken into account.
- Risk factors: Danger to human, workload amplitude and frequency or ergonomic factors may be of relevance.
- Assumptions about the human worker: A worker profile could be generated that maps to a cost function, using properties such as attention level, general experience level, and reliability.

A cost function for a robot could e.g. look like  $c_r(w, a) = \omega_1 f_t(w, a) + \omega_2 f_p(w, a)$ , where  $w$  and  $a$  denote the worker-action pair of the assignment,  $f_t$  the amount of time  $w$  would

need to perform  $a$  and  $f_p$  the amount of power consumption of  $w$  when performing  $a$ .  $\omega_1$  and  $\omega_2$  are weighting factors. A human cost function could be  $c_h(w, a) = \omega_1 f_a(w, a) + \omega_2 f_w(w, a)$ , where  $f_a$  is a measurement for the anticipated attention level of the human when performing action  $a$  and  $f_w$  is a workload measurement. Note that joint costs, i.e. costs that arise from interactions, can be treated explicitly. This is useful in order to integrate interdependencies between two specific agents into the cost function.

Now, in order to make use of the advantages of the  $A^*$  algorithm, a suitable heuristic needs to be defined. The approach for the stated search problem is as follows. The minimum amount of assembly actions  $n_{a,\min}$  that need to be applied under the assumptions from Sec. 2.3.1 to get from the current state  $\zeta$  to the goal state  $\zeta_g$  are found to be

$$n_{a,\min} = \log_2(\max_i(n_\rho(v_{o,i}))). \quad (2.20)$$

The function  $n_\rho$  yields the number of parts of the subassembly  $\Gamma_s$  that corresponds to the OR node  $v_{o,i}$ . An admissible and consistent (or monotone) heuristic can then be derived by multiplying  $n_{a,\min}$  with the minimum cost an agent can achieve for any action that has not yet been applied. If the heuristic is admissible and consistent the algorithm will find the optimal path [311].

**4) Problem Reduction** Due to the large number of possible agent assignments to actions if many agents are used, or a complex assembly with many possibilities is to be built, it is often more efficient to reduce the problem complexity first. The proposed method to achieve this, is to reduce the search space by simplifying the AND/OR graph by separating it into multiple smaller graphs, i.e. independent problems. For this, reducible subassemblies are introduced.

**Definition 4: Reducible Subassembly** Let  $v_{o,R}$  be the root node of the partial AND/OR graph  $Y_R \subset Y_A$  that is induced by the reducible subassembly  $\Gamma_{s,R}$  and  $V_{a,p}$  the set of parent nodes of  $v_{o,R}$ . If all edges  $E = \{(v_{o,R}, v_o | v_o \in V_{a,p})\}$  were removed, the result would be two distinct AND/OR graphs, i.e.  $Y_R \cup Y'_A = \emptyset$ , where  $Y'_A = Y_A \setminus Y_R$ . Obviously, this scheme is a divide-and-conquer approach, where the actions that represent the removed edges are ignored at first. The graph  $Y'_A$  is now smaller because it only contains the root node  $v_{o,r}$  of the reducible subassembly  $P_{s,R}$ . Hence, the allocation problem is easier to solve. This is the case because several small search spaces are now present instead of a single larger one.

### 2.3.3.2 Agent-Level Task Allocation

The agent level implements the autonomous behavior of a single robot to ensure the safe and successful execution of the actions that are assigned to the robot by the planner at team level. The assembly skills at agent level directly correspond to the actions received from the team-level planner. Arrow IV in Fig. 2.3 depicts this connection. The GGTWreP framework introduced in Sec. 3.2.5 provides the necessary capabilities for the robot.

## 2.4 Machine Learning

### 2.4.1 Algorithms

In this section the choice of algorithms that were tested in the real-world experiments in Sec. 4.2 are discussed. Due to its structure black-box optimization methods are best suited for the used GGTWreP framework. Within this subclass of machine learning the number of potential algorithms was reduced by comparing the requirements with the capabilities of candidate algorithms.

Since the intention is to learn real-world physical manipulation problems, the algorithm will face various challenges that result in specific requirements:

- Generally, no feasible analytic solution is available, meaning simple models cannot be relied upon.
- Gradients are usually not available.
- Real-world problems are inherently stochastic due to various factors such sensor noise, actuator noise, model inaccuracies and noise coming from the task environment itself (e.g. wear and tear over time).
- No assumptions are possible on minimum or cost function convexity.
- Violation of safety, task or quality constraints have to be considered which means that discontinuities in the cost function are possible due to sudden failure events. In general this can be stated as the requirement to handle unknown constraints.
- Low computational effort is required. This is essential for real-world robot learning with limited computational resources (due to e.g. lack of network connection, low desired power consumption of the system, etc.)
- Low total learning time is desired which is a major factor for real-world robot learning which cannot be scaled up as easily as a simulation.

Thus, suitable learning algorithms must provide a numerical black-box optimization and cannot rely on gradients. Also, stochasticity must be considered and the method has to be able to optimize globally. Furthermore, it should handle unknown and noisy constraints and finally, it must use only few computational resources while still keeping the number of required samples low.

Table 2.1 lists several groups of state-of-the-art optimization methods and compares them with respect to above requirements. In [312] a similar reasoning can be found for a humanoid walking problem. Note that extensions exist for the stochastic case for all of the algorithms.

Generally, gradient-descent-based algorithms are unsuitable since they rely on gradients and do not optimize globally. Grid search and evolutionary algorithms cannot handle unknown constraints very well without extensive knowledge about the problem they optimize, i.e. make use of well-informed barrier functions. The latter aspect applies also to particle swarm algorithms. Only Bayesian optimization (BO) in accordance to [313] is capable of explicitly handling unknown noisy constraints during optimization. However,

**Table 2.1:** Suitability of existing learning algorithms with respect to the desired properties

Method	no gradient	stochasticity assumption	global optimizer	unknown constraints	low computational requirements
Grid Search	+	+	+	−	+
Gradient-descent family	−	+	−	−	+
Evolutionary algorithms	+	+	+	−	+
Particle Swarm	+	+	+	−	+
Bayesian Optimization	+	+	+	+	−

usually the best performing implementations of Bayesian optimization have high computational requirements if they run for a higher number of trials. Judging from the table all algorithms except gradient descent seem more or less suited for real-world robot learning, thus they are evaluated in an experimental setting.

In later experiments also the hierarchical relative entropy search (HiREPS) algorithm and a newly developed SVM-based optimization method which partitions the parameter space to find an optimal (and robust) solution were compared.

#### 2.4.1.1 Bayesian Optimization

In general, BO finds the minimum of an unknown objective function  $f(\boldsymbol{\theta})$  on some domain  $\mathbb{D}$  by developing a statistical model of  $f(\boldsymbol{\theta})$  where  $\boldsymbol{\theta}$  is the set of parameters. Apart from the cost function, it has two major components, which are the prior and the acquisition function.

**Prior** A Gaussian process is used as prior to derive assumptions about the function being optimized. The Gaussian process has a mean function  $m : \mathbb{D} \rightarrow \mathbb{R}$  and a covariance function  $B : \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{R}$ . As a kernel the automatic relevance determination (ARD) Matérn 5/2 kernel is used which is given by

$$B_{M52}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \theta_0(1 + \sqrt{5r^2(\boldsymbol{\theta}, \boldsymbol{\theta}')} + \frac{5}{3}r^2(\boldsymbol{\theta}, \boldsymbol{\theta}')e^{-\sqrt{5r^2(\boldsymbol{\theta}, \boldsymbol{\theta}')}}), \quad (2.21)$$

with

$$r^2(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{i=1}^d \frac{(p_i - p'_i)^2}{\theta_i^2}. \quad (2.22)$$

This kernel results in twice-differentiable sample functions which makes it suitable for practical optimization problems as stated in [314]. It has  $d + 3$  hyperparameters in  $d$  dimensions, i.e. one characteristic length scale per dimension, the covariance amplitude  $\theta_0$ , the observation noise  $\nu$  and a constant mean  $m$ . These kernel hyperparameters are integrated out by applying Markov chain Monte Carlo (MCMC) via slice sampling [315].



**Acquisition function** Predictive entropy search with constraints (PESC) is used as a means to select the next parameters  $\boldsymbol{\theta}$  to explore, as described in [316].

### 2.4.1.2 Particle Swarm Optimization

Usually particle swarm optimization starts by initializing all particle positions  $\mathbf{x}_i(0)$  and velocities  $\mathbf{v}_i(0)$  with a uniformly distributed random vector, i.e.  $\mathbf{x}_i(0) \sim U(\mathbf{b}_{\text{lb}}, \mathbf{b}_{\text{ub}})$  and  $\mathbf{v}_i(0) \sim U(-|\mathbf{b}_{\text{ub}} - \mathbf{b}_{\text{lb}}|, |\mathbf{b}_{\text{ub}} - \mathbf{b}_{\text{lb}}|)$  with  $U$  being the uniform distribution. The particles are evaluated at their initial positions and their personal best  $\boldsymbol{\theta}_i^*$  and the global best  $\boldsymbol{\theta}^*$  are set. Then, until a termination criterion is met, following steps are executed:

1. Update particle velocity:

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1(\boldsymbol{\theta}_i^* - \mathbf{x}_i(t))\mathbf{R}_1 + \mathbf{v}_i(t) + c_2(\boldsymbol{\theta}^* - \mathbf{x}_i(t))\mathbf{R}_2 \quad (2.23)$$

where  $\mathbf{R}_1$  and  $\mathbf{R}_2$  are diagonal matrices with random numbers generated from a uniform distribution in  $[0, 1]$  and  $c_1, c_2$  are acceleration constants usually in the range of  $[0, 4]$ .

2. Update the particle position:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (2.24)$$

3. Evaluate the fitness of the particle  $f(\mathbf{x}_i(t+1))$ .
4. Update each  $\boldsymbol{\theta}_i^*$  and global best  $\boldsymbol{\theta}^*$  if necessary.

### 2.4.1.3 Covariance Matrix Adaptation Evolution Strategy

The covariance matrix adaptation evolution strategy (CMA-ES) is an optimization algorithm from the class of evolutionary algorithms for continuous, non-linear, non-convex black-box optimization problems [317, 318].

The algorithm starts with an initial centroid  $\mathbf{c}_0 \in \mathfrak{R}^n$ , a population size  $\lambda$ , an initial step-size  $\sigma > 0$ , an initial covariance matrix  $\mathbf{C} = \mathbf{I}$  and isotropic and anisotropic evolution paths  $p_\sigma = 0$  and  $p_c = 0$ .  $\mathbf{c}_0$ ,  $\lambda$  and  $\sigma$  are chosen by the user. Then the following steps are executed until the algorithm terminates.

1. Evaluation of  $\lambda$  individuals sampled from a normal distribution with mean  $\mathbf{c}$  and covariance matrix  $\sigma\mathbf{C}$ .
2. Update of centroid  $\mathbf{c}$ , evolution paths  $p_\sigma$  and  $p_c$ , covariance matrix  $\mathbf{C}$  and step-size  $\sigma$  based on the evaluated fitness.

### 2.4.1.4 Latin Hypercube Sampling

Latin hypercube sampling (LHS) [319] is a method to sample a given parameter space in a nearly-random way. In contrast to pure random sampling LHS generates equally distributed random points in the parameter space. Such a method is useful in the present case to see whether systematic checking of the parameter space might come close to already good solutions without actually learning. It can be an indication as to how necessary learning actually is for the particular problem cases.

### 2.4.1.5 HiREPS

The Hierarchical Relative Entropy Search algorithm was proposed in [320]. Here, its function is only briefly described.

As input the algorithm takes an information loss tolerance  $\epsilon$ , an entropy tolerance  $\kappa$ , and a number of options  $n$ . It initializes the policy  $\pi$  with  $n$  Gaussians with random means. Then, for a number of  $L$  episodes it executes the following steps:

1. The sample policy is set to

$$q(a|s) = \sum_{\mathcal{O}} \pi_{\text{old}}(\mathcal{O}|s) \pi_{\text{old}}(a|s, \mathcal{O}) \quad (2.25)$$

2. Then new samples are collected from the sample policy and added to the current dataset by

$$\{s_j, p(s_0), a_j, q(a|s_j), R_j\}_{j \in \{1, \dots, N\}} \quad (2.26)$$

3. The importance weights are calculated to

$$v_i^k = \frac{q_k(s_i, a_i)}{\sum_{h=k-H}^k q_h(s_i, a_i)} \forall i \quad (2.27)$$

4. The proposal distribution is updated by

$$\tilde{p}(\mathcal{O}|s_i, a_i) = p_{\text{old}}(\mathcal{O}|s_i, a_i) \forall i \quad (2.28)$$

5. The dual function is minimized by

$$[\theta, \nu, \epsilon] = \arg \min_{[\theta, \nu, \epsilon]} g(\theta, \nu, \epsilon) \quad (2.29)$$

6. Finally, the policy can be updated by

$$p(s_i, a_i, \mathcal{O}) = v_i^k \tilde{p}(\mathcal{O}, s_i, a_i)^{1+\frac{\epsilon}{\nu}} \exp\left(\frac{R_i - \theta^T \phi(s_i)}{\nu}\right) \quad (2.30)$$

and  $\pi(\mathcal{O}|s)$  and  $\pi(a|s, \mathcal{O})$  can be estimated. The output is a new policy  $\pi(a, \mathcal{O}|s)$ .

### 2.4.1.6 Parameter Space Partitioning Algorithm

The parameter space partitioning (PSP) algorithm [321] runs for  $k$  episodes. Each episode consists of a number of trials  $n_e$ . For each trial  $i$  of an episode, parameters  $\theta_{s_i}$  are sampled  $\in q(a)$  in sample-space, i.e. the hypercube, with  $q(a)$  being the sampling policy. These are translated into solution-space and applied to the optimization problem. The resulting reward  $r_i$  is stored together with the parameters in sample-space  $\theta_{s_i}$ . If a trial

is unsuccessful, the reward  $r_i$  is set to a negative value,  $r_i = -1$ . This is done to assure negative classification in the update step. At the end of each episode, the sampling policy  $q(a)$  is updated. The sampling policy  $q(a)$  consists of two elements, a proposal policy  $p(a)$  and a filtering policy  $f(p(a))$ .  $p(a)$  proposes parameters until a sample has been found which is accepted by the filtering policy  $f(p(a))$ .  $p(a)$  proposes parameters  $\theta_p$ , which are then classified by the filtering policy  $f(\theta_p)$ . The filtering policy is a non-linear support vector machine (SVM).

**Proposal Policy** In the beginning, the proposal policy is a Latin hypercube sampler since there is not yet enough data to generate meaningful data. Instead the available solution space is more evenly sampled. After the first episode a uniform random sampler is used. In later episodes, assuming sufficient data is available, a Gaussian mixture model (GMM) is used as policy.

**Filtering Policy** The filtering policy is a non-linear SVM with rbf-kernels. It is only used if a sufficient number of successful samples (in the sense of a successful skill execution) is available to ensure a robust estimation.

## 2.4.2 Performance Metrics

In this section the metrics used for learning and specifically transfer learning are introduced.

### 2.4.2.1 Monotonically Decreasing Cost Function

The monotonically decreasing cost function  $\mathcal{Q}_{c,m}$  always has the lowest already seen cost value at any particular time point. It gives a better impression of the learning progress than a pure cost function  $\mathcal{Q}_c$ . It is defined as

$$\mathcal{Q}_{c,m_i} = \begin{cases} \mathcal{Q}_{c_{i-1}} & \text{if } \mathcal{Q}_{c_i} > \mathcal{Q}_{c_{i-1}} \\ \mathcal{Q}_{c_i} & \text{else} \end{cases} \quad (2.31)$$

Figure 2.7 provides a visual aid for it.

### 2.4.2.2 Average Cost and Confidence Interval

A single learning process is strongly stochastic and can significantly vary from process to process. Therefore, learning a specific skill (i.e. a specific experiment) is repeated to collect enough data to derive statistically sound results. After repeating an experiment for  $n$  times  $n$  data sets  $\mathcal{D}_i$  are available. Now, the monotonically decreasing cost function either based on trials or time can be averaged over the  $n$  data sets. Also a confidence interval can be calculated from the data [322]. To calculate a confidence interval  $i_c$  one has to first select a confidence level (e.g. 90% or 95%). Based on that the Z-value  $Z$  can be taken from a Z-table and the formula

$$i_c = Z \frac{s}{\sqrt{n}} \quad (2.32)$$

can be applied, where  $s$  is the standard deviation and  $n$  the number of samples. Thus, an average cost can be defined with the confidence interval given by

$$\bar{Q}_c \pm Z \frac{s}{\sqrt{n}}. \quad (2.33)$$

### 2.4.2.3 Average Learning Success Rate

The results are not only analyzed in terms of the optimization goal and how fast learning converges to an optimum, but also in terms of actual success, i.e., how well the learning algorithm manages to find feasible areas in the parameter space. Therefore, the average learning success rate (ALSR) is defined to be

$$\text{ALSR}(t) = \frac{1}{n} \sum_{l=1}^{n_e} Q_{s,l}(t) \quad (2.34)$$

where  $n_e$  denotes the number of repeated experiments to learn the same skill and  $Q_{s,l}(t)$  is the success indicator whether the trial at time  $t$  for experiment  $l$  was successful or not. This gives a statistical measure of how successful a skill is over time during learning.

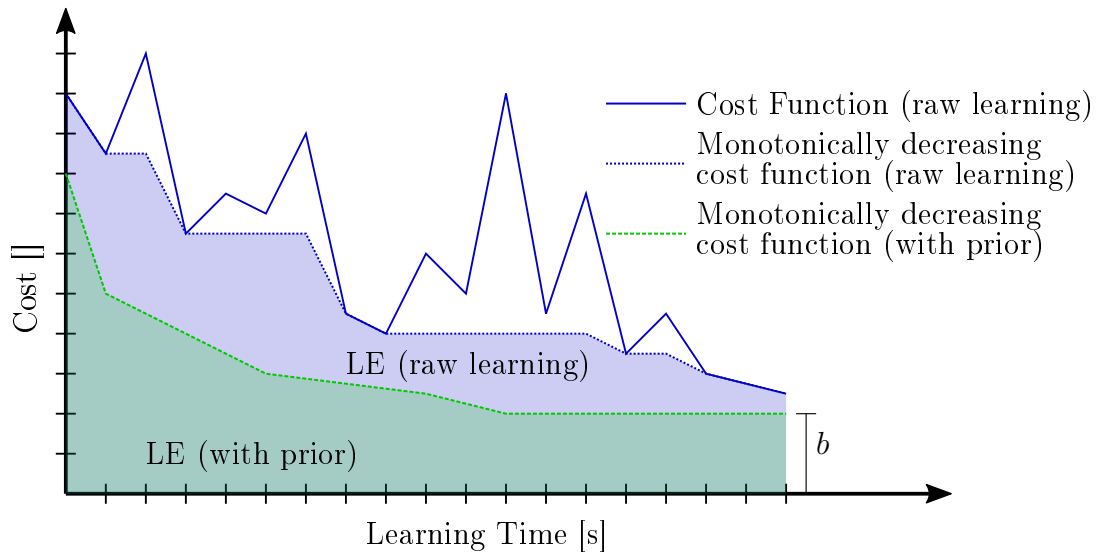
### 2.4.2.4 Learning Effort and Transferability

To interpret the results of the transfer learning experiments in Sec. 4.2.3, a comparable quantification of the achieved transfer learning performance is required. The highly non-linear and noisy nature of real-world robot manipulation learning makes standard existing metrics from system theory or image classification, such as MNIST or CIFAR, unsuitable. To the best of the author's knowledge, there are no metrics that explicitly analyze transfer learning in robot manipulation. There are some in the domain of image classification, such as LEEP [323] or H-score [324], but the problem at hand is fundamentally different since dynamic processes are compared. Existing metrics for comparing two systems with each other, such as  $\mu$ -gap [325, 326], usually focus on linear or non-statistical problems. Statistical distance measures like the Bhattacharyya distance [327], K-S-Test [328], or the KL Divergence [329] can be useful in constructing an evaluation metric. However, they are more difficult to interpret since they do not directly reflect the learning process itself. Thus, to analyze the learning results quantitatively, three metrics are introduced: learning effort (LE), learning effort ratio (LER) and empirical transferability (ET). These metrics are designed to capture the performance of the knowledge transfer over the entire experiment, as is detailed below.

Specifically, the learning effort LE is defined as

$$\text{LE} = \int_{\mathfrak{D}} Q_{c,m}(t) dt, \quad (2.35)$$

where  $\mathfrak{D}$  is the available sample data of the learning process in chronological order containing a tuple  $(\theta, Q_{c,m})$  with chosen parameters and a resulting cost function for every sample  $k$ . The quantity  $Q_{c,m}(t)$  denotes the value of the monotonically decreasing cost function at time  $t$ . In consequence, LE measures the minimum total cost required to learn a skill.



**Figure 2.7:** Monotonically decreasing cost function, learning effort LE, and learning effort ratio LER

The learning effort ratio LER is the ratio between the LEs from data sets  $\mathcal{D}_i$  and  $\mathcal{D}_j$

$$\text{LER} = \frac{\text{LE}_i - b}{\text{LE}_j - b}, \quad (2.36)$$

where the factor  $b$  is defined as  $\min(\min(\mathcal{Q}_{c,m,i}(t)|\mathcal{D}_i|), \min(\mathcal{Q}_{c,m,j}(t)|\mathcal{D}_j|))$ . It is the lowest value of  $\mathcal{Q}_{c,m}(t)$  in either  $\mathcal{D}_i$  or  $\mathcal{D}_j$ , and it is subtracted to remove the bias from the data that would otherwise be caused by the minimum reachable cost. The LER can be used to quantify the effectiveness of using prior knowledge from task  $\iota_j$  on task  $\iota_i$ . Raw learning, i.e., learning  $\iota_i$  without prior knowledge, is represented by the dataset  $\mathcal{D}_i$ , and learning  $\iota_i$  with prior knowledge from  $\iota_j$  is represented by  $\mathcal{D}_j$ .

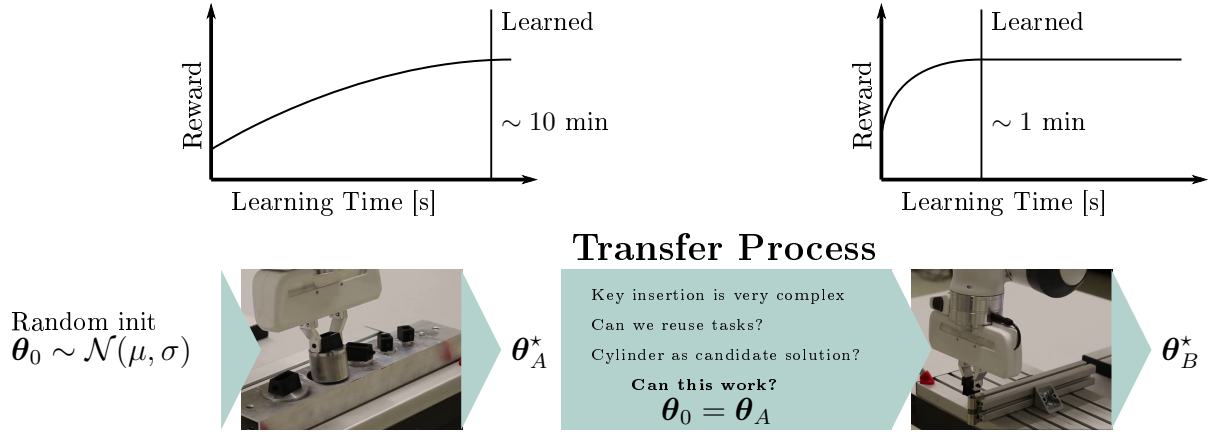
Finally, the empirical transferability ET is defined as

$$\text{ET}_{i,j} = \min\left(\frac{\text{LER}_{i,i}}{\text{LER}_{i,j}}, 1\right). \quad (2.37)$$

It relates the LER into the context of task  $t_i$  by normalizing it. This is done by using  $\text{LER}_{i,i}$ , which is the LER of learning task  $t_i$  with prior knowledge from itself. The underlying assumption is that, theoretically, this is the most effective knowledge transfer. Any  $\text{LER}_{i,j}$  for any task  $\iota_j$  can then be seen with respect to the best-case scenario, which essentially amounts to a measured transferability between the two tasks. The ET cannot grow larger than 1.

### 2.4.2.5 Learning Speedup

The learning speedup is defined as the reduction in the learning time when using prior knowledge compared to raw learning with an uninformed initialization. There is no practical way to determine the exact speedup a priori, since there is no reliable objective criterion for when a real-world skill is learned. For example, a task can be considered learned if it reaches a prespecified cost value or as soon as the cost changes remain within a predefined confidence interval. In this work, the latter is applied.



**Figure 2.8:** Transfer learning observation with a 10x reduction in the learning time for task B.  $\theta_0$  denotes the initial parameters drawn from a normal distribution  $\mathcal{N}(\cdot)$  with a mean  $\mu$  and standard deviation  $\sigma$ .  $\theta_A^*$  and  $\theta_B^*$  are the optimal parameters for tasks A and B, respectively.

### 2.4.3 Robot Motor Memory Effect

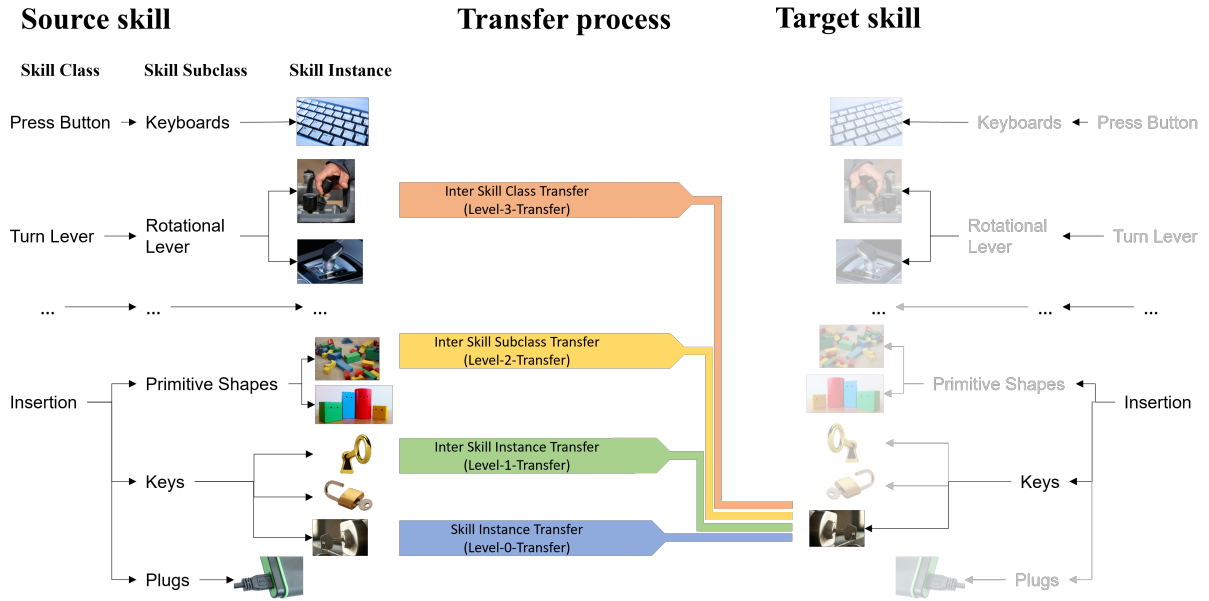
In a randomly performed demonstration of the tactile skill formalism GGTWreP (see Sec. 3.1), a significant speedup of roughly 10 $\times$  was observed when simply reusing learned parameters from a cylinder insertion in a key insertion task [330], leading to learning times of 1 minute instead of 10 minutes, see Fig. 2.8.

In hindsight, it is hypothesized that the specific task-informed design of the tactile skill learning framework, which heavily depends on well-known principles from human motor control, made this straightforward transfer possible. This somewhat surprising finding encouraged further investigations into the effects of reusing knowledge from seemingly similar skills, which is also believed to be useful for further theoretical groundwork.

Based on this initial experimental observation, it is believed that a transfer effect for tactile skills is embedded in the developed structure. This allows us to exploit inherent similarities between skills for faster learning, even without specially designed complex computational transfer mechanisms. The confirmation of such a behavior could be interpreted as an emerging robot motor memory (RMM) that plays a similar role as the motor memory effect in human motor control. However, no claim is made here that there is a connection between the two. Although the similarity between two skills as a whole may not be exactly quantifiable by experimental work, transferability might provide a way to “sample” the similarity locally. In other words, if two tactile skills are similar enough, a learned optimal policy solution for the first skill would speed up the time taken to find an optimal policy solution for the second skill.

To structure and analyze skill similarities and achievable transfer learning, the taxonomy classifications *skill class*, *skill subclass*, and *skill instance* from Sec. 2.1.3 are used (see Fig. 2.9). This classification results in level-2, level-1, and level-0 transfers, respectively.

A transfer learning process is denoted as level-0 between different tasks that belong to the same skill instance, level-1 from a *skill instance* to another instance from the same *skill sub class*, level-2 from a different *skill sub class*, and level-3 from another *skill class*. In this work level-0, level-1, and level-2 transfers are addressed. A level-0 transfer refers to, e.g., a cylinder insertion process in which the task was learned with prior knowledge from its own previous learning. Level-1 indicates that this cylinder was learned with knowledge



**Figure 2.9:** Hierarchical three-level classification proposal of tactile manipulation skills

from a different cylinder. In a level-2 transfer, the cylinder was successfully learned based on the knowledge from, e.g., a key insertion skill. Based on these definitions and the speedup metric introduced in Sec. 2.7 a hypothesis can be stated that on average across all investigated tasks the speedup factor is highest for level-0 transfers. Level-1 transfers are second and level-2 transfers have the lowest speedup. This also means that there is a learning effort ratio (LER) that increases with the transfer level.

## 2.5 Conclusion

This chapter introduces the theoretical foundations for the learning architecture in Ch. 3. The concept of tactile skills is developed and connected to a formal process definition through a hierarchical taxonomy. These components are then used in a synthesis pipeline that automatically selects a tactile policy to solve a given input process. A suitable planning system is described that uses tactile skills in sequences to solve an assembly problem for a collaborative team of humans and robots. A number of learning algorithms are introduced that are later experimentally evaluated using a number of performance metrics. Finally, a robot motor memory effect is described that forms the basis for the transfer learning capabilities of the learning architecture.





# 3

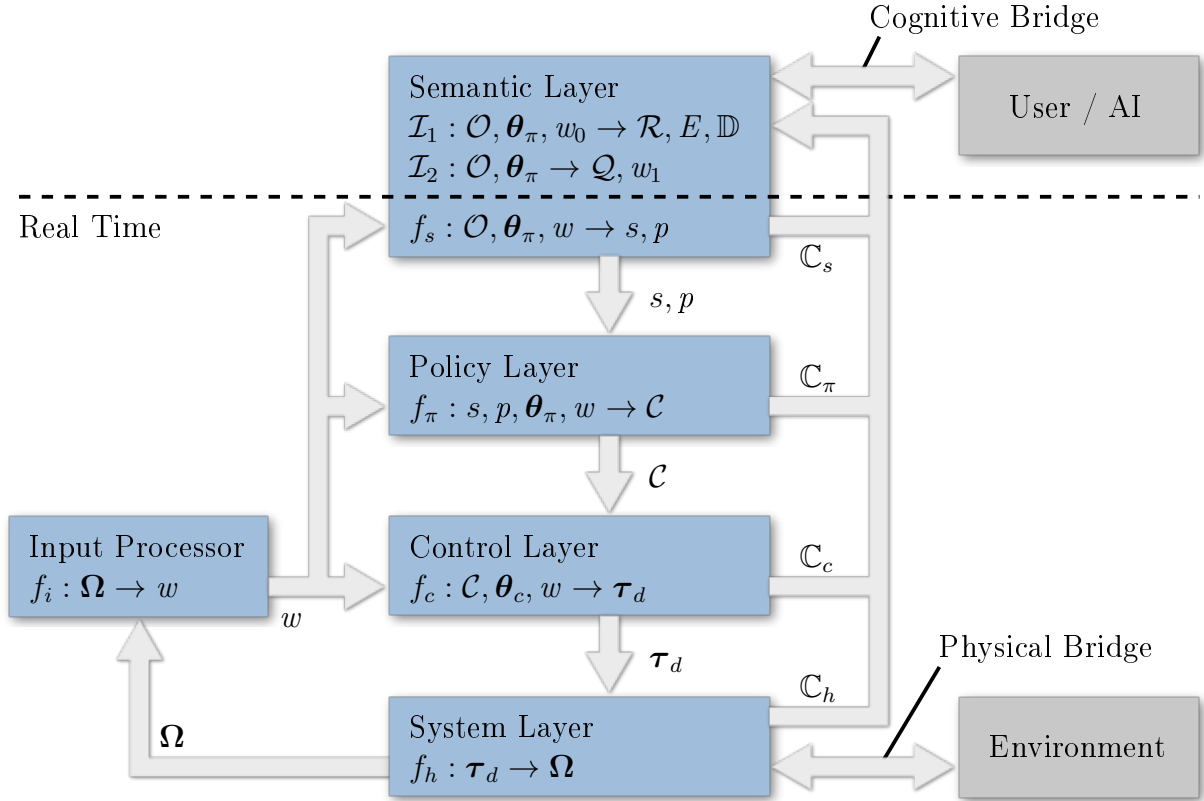
## System Architecture and Validation Cases

This chapter describes the overall learning architecture that realizes the theoretical foundations provided in Ch. 2. It is used for the experimental work described in Ch. 4. First, in Sec. 3.1 the graph-guided twist-wrench policy approximation (GGTWreP) framework is introduced. It is a multi-layered architecture that models the tactile skill concept. It consists of a system layer connected to the tactile platform, a control layer that provides a tactile controller, a policy layer that implements a tactile policy, and a semantic layer that handles process steps and connects to a learning algorithm. Section 3.2 outlines the Machine Intelligence Operating System (MIOS), a highly integrated software stack that efficiently implements the GGTWreP framework in a scalable and distributed fashion. Section 3.3 describes a number of validation experiments and applications where the architecture developed in this thesis has been successfully used. These cover the field of dentronics, a telepresence application, a collaborative assembly station, and a highly sophisticated robotic art project in the Pinakothek der Moderne in Munich, Germany. Section 3.4 presents two more cases in which the learning component was the main matter of interest. These are a manipulation learning setup for industrial-grade peg-in-hole and a complex, distributed robots system denoted the collective. This chapter was written based on [15, 25, 331, 332].

### 3.1 GGTWreP Framework

This section introduces the graph-guided twist-wrench policy approximation (GGTWreP) framework [23, 24]. It consists of multiple hierarchical layers. Each layer models a different aspect of tactile manipulation. This multi-layer structure descends from a high-level semantic layer down to the hardware system layer that is directly connected to the physical robot platform, which is coupled to the real world (see Fig. 3.1).  $w \in \mathcal{W}$  denotes an element of the world state space  $\mathcal{W}$  and contains, e.g., the robot poses, external forces, or object positions.  $\Omega \in \mathcal{W}$  denotes the percept vector, and it contains information received by internal or external sensors (see also Sec. 2.1).

**Layers** The framework layers are described in detail in the sections below. Each layer receives inputs and additional parameters from the above layer, and provides outputs to the below layer. They also provide constraints  $\mathbb{C}$  in terms of the context of the task and the system's limits. These constraints model the limits of the valid input of the respective layer (e.g., the maximum admissible velocities). Additionally, the percept processor updates and provides the world state  $w$  to the other components based on the percept vector  $\Omega$  and internal models. Figure 3.1 provides an overview of the GGTWreP framework with its different layers.



**Figure 3.1:** Architectural overview of the GGTWreP framework

- The semantic layer decides the currently active high-level state as well as the currently active policy in the policy layer. Furthermore, it is the point of communication with higher-level components such as learning and planning algorithms.
- The policy layer holds a set of ordinary differential equations (ODE), which are embedded in a graph structure, that produce tactile commands.
- The control layer implements a unified impedance / force controller that is fed by tactile commands sent from the policy layer. It then produces desired motor commands for the system layer. Also, safety mechanisms ensure the system and process constraints.
- The system layer is the lowest layer, and it sends motor commands from the control layer to the robot hardware. It also provides the current robot state to the other layers.

**Objects** A skill is instantiated by using objects  $\mathcal{O}$  that define the environment that is relevant to the skill, similar to the definition of manipulation processes introduced in Sec. 2.1.2. Note that all skills also contain the end effector (*End Effector*) as a default object. It has the handle EE.

**Semantic Layer** The semantic layer consists of a non-deterministic (non-realtime) component for interface purposes and a deterministic (realtime) component that contains a hierarchical state machine.

The non-deterministic (non-realtime) interface consists of two parts: an execution interface  $\mathcal{I}_1$  and a learning interface  $\mathcal{I}_2$ , which are represented by the functional mappings

$$\begin{aligned}\mathcal{I}_1 &: \mathcal{O}, \boldsymbol{\theta}_\pi, w_0 \rightarrow \mathcal{R}, E, \mathbb{D}, \\ \mathcal{I}_2 &: \mathcal{O}, \boldsymbol{\theta}_\pi \rightarrow \mathcal{Q}, w_1.\end{aligned}$$

$\mathcal{R} \in \mathcal{W}$  describes the requirements in terms of the world state for executing the skill,  $E \in \mathcal{W}$  is the effect of the skill when it is applied to a user-defined world state  $w_u$  with objects  $\mathcal{O}$ , and  $\mathbb{D}$  is the domain of valid parameters for every  $\theta \in \boldsymbol{\theta}$ .

The deterministic component contains a discrete two-layer state machine that consists of four high-level states; namely, the initial state  $s_0$ , the policy state  $s_\pi$ , the error state  $s_e$ , and the final state  $s_1$ .  $s_0$  is active in the beginning,  $s_1$  is active at the end in a nominal case,  $s_e$  is the end state in case an error occurs, and  $s_\pi$  activates the policy layer. Three transitions govern the switch behavior at the top level of the state machine. They directly implement the boundary conditions from the process definition introduced in Sec. 2.1.2, therefore the explanation of their meaning is omitted here. However, there are a number of default conditions coming from the robot system:

- There is a default precondition  $\mathcal{C}_{\text{pre},0} = \{T_{\text{EE}} \in \text{ROI}\}$  which states that the robot has to be within a suitable region of interest (ROI).
- There are three default error conditions  $\mathcal{C}_{\text{err},0} = \{|\mathbf{f}_{\text{ext}}| > \mathbf{f}_{\text{ext,max}}, T_{\text{EE}} \notin \text{ROI}, t > t_{\text{max}}\}$  which state that the robot may not exceed the maximum external forces, may not leave the ROI, and may not exhaust a maximum time for skill execution.

The policy state  $s_\pi$  contains a state machine layer that is denoted as manipulation graph, and it implements the execution state from the process definition. The graph may be described by  $G(P, \Delta)$ , where  $P$  denotes the set of policy approximators (nodes), and  $\Delta$  denotes the set of transitions (edges). The transitions are conditions that, if true, trigger a switch of the current PA according to the graph structure.

The deterministic component of the semantic layer is represented by the functional mapping

$$f_s : \mathcal{O}, \boldsymbol{\theta}_\pi, w \rightarrow s, p. \quad (3.1)$$

**Policy Layer** The policy layer contains a set of sets (which are denoted policy approximators) of ODEs. Each policy approximator (PA) implements one process state while maintaining the stated conditions. The currently active PA  $p$  is determined by the semantic layer.

A single PA implements a tactile policy  $\boldsymbol{\pi}_d$ .

Overall, the policy layer functional mapping is expressed by

$$f_\pi : s, p, \boldsymbol{\theta}_\pi, w \rightarrow \mathcal{C}, \quad (3.2)$$

where  $p$  and  $s$  denote the currently active PA and high level state as determined by the semantic layer. For  $s \neq s_\pi$ , a default PA

$$\mathcal{C} = \begin{bmatrix} \dot{\boldsymbol{x}}_d \\ \boldsymbol{f}_d \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ [\mathbf{0}^T \ f_{\text{grasp}}]^T \end{bmatrix}, \quad (3.3)$$

is activated, where  $f_{\text{grasp}}$  denotes the current grasp force.

**Control Layer** The control layer receives commands  $\mathcal{C}$  from the policy layer and calculates the desired motor commands  $\boldsymbol{\tau}_d$ . It uses controllers such as the ones introduced in Sec. 2.1. Architecturally, the control layer is encoded by the functional mapping

$$f_c : \mathcal{C}, \boldsymbol{\theta}_c, w \rightarrow \boldsymbol{\tau}_d. \quad (3.4)$$

Furthermore, the control layer contains safety mechanisms such as value and rate limitations, collision detection, reflexes, and virtual walls.

**System Layer** The system layer is expressed by the functional mapping

$$f_h : \boldsymbol{\tau}_d \rightarrow \boldsymbol{\Omega}. \quad (3.5)$$

It defines the control/sensing interface to the robot's hardware system and other devices. It encapsulates any subsequent hardware-specific control loops and other processes. It is the implementation of the *Tactile Platform*.

**Input Processor** The input processor holds all of the models for internal and external processes. Examples of internal models are the estimated mass matrix  $\hat{\boldsymbol{M}}(\boldsymbol{q})$ , Coriolis forces  $\hat{\boldsymbol{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ , and gravity vector  $\hat{\boldsymbol{g}}(\boldsymbol{q})$ . External models describe the state of environmental elements such as the physical objects handled by the robot. For example, if the robot were to place an object at a new location, a model of the object would be updated with the new pose. The processor continuously updates the models by using  $\boldsymbol{\Omega}$  from the current world state  $w$ . The processor functional mapping is

$$f_i : \boldsymbol{\Omega} \rightarrow w. \quad (3.6)$$

**Task Frame** The task frame  $T$  defines a coordinate frame relative to the robot's origin frame  $O$  so that there exists the transformation  ${}^O T_T$ . All skill commands  $\mathcal{C}$  are then calculated in the task frame and transformed via this transformation matrix into the frame of origin.

### 3.1.1 Implementation Examples

In this section, two examples of processes and their implementation as GGTWreP models are presented. The examples are inserting an Ethernet plug and cutting a piece of cloth. Figure 3.2 visualizes the process and the skill implementation. The details of the policy selection through  $\mathcal{T}$  are described in Sec. 2.2 with the same process examples.

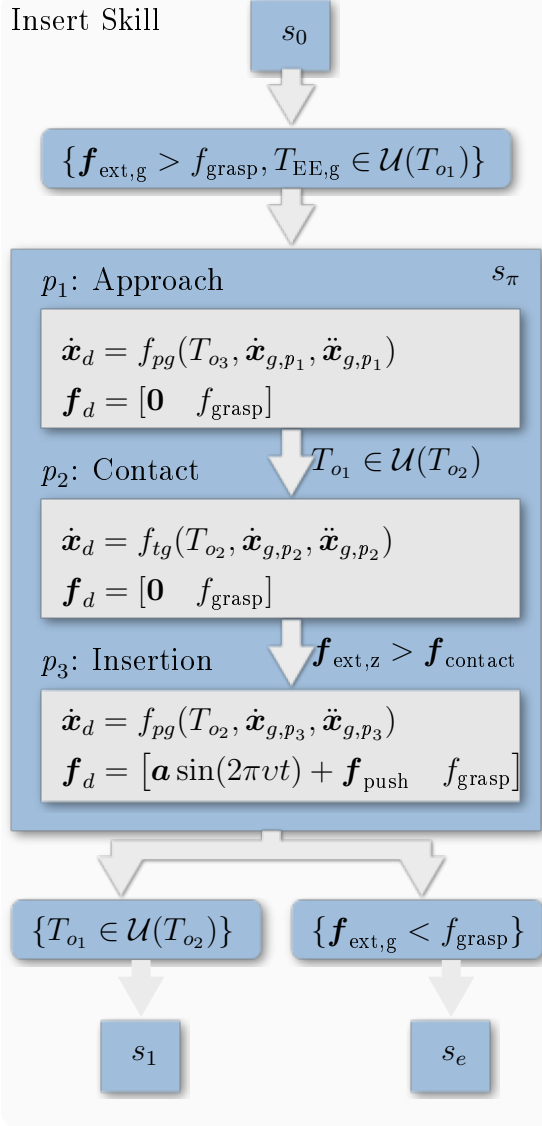
## Insert Process

$$\begin{aligned}
\mathcal{O} &= \{o_1, o_2, o_3\} \\
\mathcal{C}_{\text{pre}} &= \emptyset \\
\mathcal{C}_{\text{err}} &= \emptyset \\
\mathcal{C}_{\text{suc}} &= \{T_{o_1} \in \mathcal{U}(T_{o_2})\} \\
\Delta &= \{T_{o_1} \in \mathcal{U}(T_{o_2}), \mathbf{f}_{\text{ext},z} > \mathbf{f}_{\text{contact}}\}
\end{aligned}$$

## Cut Process

$$\begin{aligned}
\mathcal{O} &= \{o_1, o_2, o_3, o_4, o_5\} \\
\mathcal{C}_{\text{pre}} &= \emptyset \\
\mathcal{C}_{\text{err}} &= \{\mathbf{f}_{\text{ext},z} < \mathbf{f}_{\text{cut}}\} \\
\mathcal{C}_{\text{suc}} &= \{T_{o_1} \in \mathcal{U}(o_5)\} \\
\Delta &= \{T_{o_1} \in \mathcal{U}(o_2), \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{cut}}, \\
&\quad T_{o_1} \in \mathcal{U}(o_4)\}
\end{aligned}$$

## Insert Skill



## Cut Skill

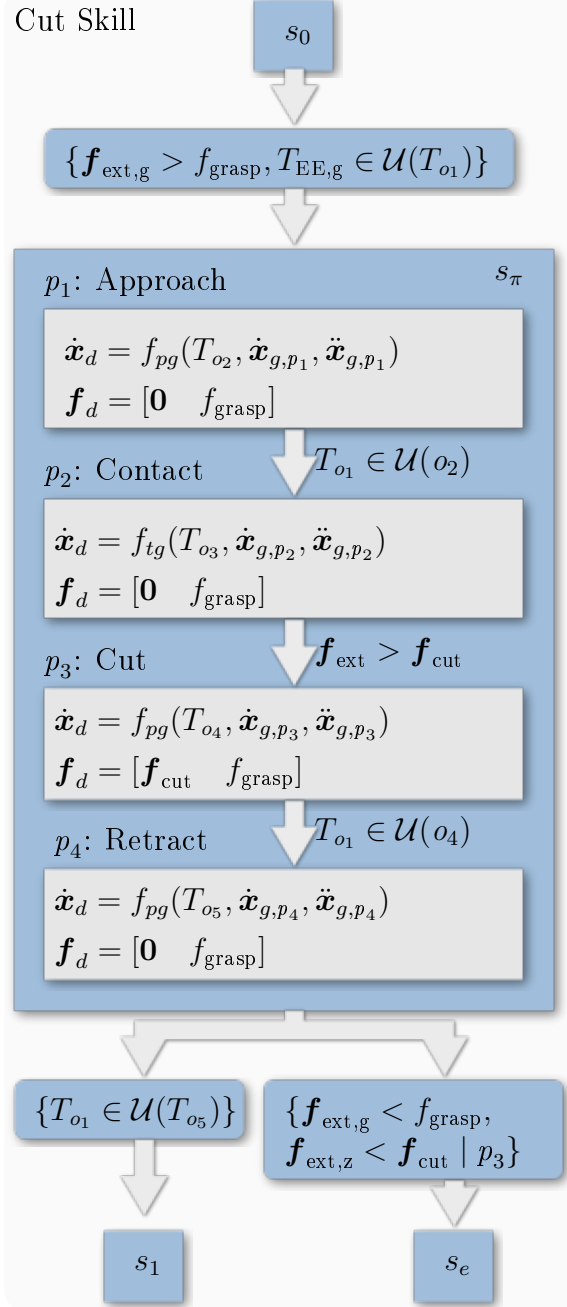


Figure 3.2: The two examples show the process description and their corresponding GGTWrep models.

### 3.1.1.1 Inserting an Ethernet Plug

In general, an insertion process can be described as fitting one object into another, i.e., matching their geometries by form-close. In an industrial context, this process is required for, for example, part mating. Details about insertion-related processes are described in specialized literature such as [333], norms [334], and robotics-related publications such as [335,336]. In the following section, the details of the skill implementation that uses the GGTWreP framework are outlined.

**Process Description** The process definition states that the *Insertable*  $o_1$  has to be moved toward an *Approach* pose  $o_3$ . From there, contact is established in the direction of the *Container*  $o_2$ . Finally, the *Insertable* has to be inserted into the *Container*.

**Conditions** The process definition states no preconditions. However, there is a default precondition that the robot has to be within the user-defined ROI, and an implementation-specific precondition that the robot must have grasped the *Insertable*  $o_1$ .

The default error conditions are that the external forces and torques must not exceed a predefined threshold, the ROI must not be left, and the maximum execution time must not be exceeded. Additionally, the robot must not lose the *Insertable*  $o_1$  at any time. Note that, for simplicity, the default conditions are not shown in Fig. 3.2.

The process definition states that, to be successful,  $o_1$  has to be matched with  $o_2$ . In the implementation, this is expressed by a predefined maximum distance  $\mathcal{U}(o_2)$ .

**Tactile Policies** The insertion skill model consists of three distinct phases: 1) approach, 2) contact, and 3) insert. The approach phase uses a simple point-to-point motion generator to drive the robot through free space toward  $o_3$ . The contact phase drives the robot into the direction of  $o_2$  until contact has been established, i.e., when external forces that exceed a defined contact threshold  $f_{\text{contact}}$  have been perceived. The insertion phase attempts to move  $o_1$  toward  $o_2$  by pushing downward with a constant wrench while employing a Lissajous figure to overcome friction and material dynamics. Additionally, a simple motion generator takes care of the end effector's orientation and lateral motion toward the goal pose. A grasp force  $f_{\text{grasp}}$  is applied simultaneously to all three phases to hold  $o_1$  in the gripper.

### 3.1.1.2 Cutting a Piece of Cloth

A cutting process can be described as dividing an object into two parts by using a cutting tool such as a knife. Details about cut-related processes are described in specialized literature such as [337] and in robotics-related publications such as [338].

**Process Description** The process definition states that the *Knife*  $o_1$  has to be moved toward an *Approach* pose  $o_3$ . From there, contact is established in the direction of the *Surface*  $o_2$ . Then, the  $o_1$  is moved toward a *Goal* pose  $o_4$  while it maintains contact with the surface. Finally,  $o_1$  is moved to a final *Retract* pose  $o_5$ .

**Conditions** The process definition does not state any preconditions. However, there is a default precondition that the robot has to be within the user-defined ROI, and an implementation-specific precondition that the robot must have grasped the *Knife*  $o_1$ .

The default error conditions are that the external forces and torques must not exceed a predefined threshold, the ROI must not be left, and the maximum execution time must not be exceeded. Additionally, the robot must not lose the *Knife*  $o_1$  at any time, and  $f_{\text{ext},z} < f_{\text{contact}}$  must be maintained when moving from  $o_3$  to  $o_4$

The process definition states that, to be successful,  $o_1$  has to be moved toward  $o_5$ .

**Tactile Policies** The cutting skill model consists of four distinct phases: 1) approach, 2) contact, 3) cut, and 4) retract. The approach phase uses a simple point-to-point motion generator to drive the robot through free space toward  $o_3$ . The contact phase drives the robot into the direction of  $o_2$  until contact has been established, i.e., when external forces that exceed a defined contact threshold  $f_{\text{contact}}$  have been perceived. The cut phase moves  $o_1$  toward  $o_4$  by using a point-to-point motion generator combined with a constant downward pushing wrench. The retract phase moves  $o_1$  toward  $o_5$  by using a point-to-point motion generator. A grasp force  $f_{\text{grasp}}$  is simultaneously applied to all four phases, a to hold  $o_1$  in the gripper.

## 3.2 Machine Intelligence Operation System

### 3.2.1 System Overview

The Machine Intelligence Operating System (MIOS) has been developed as a software platform for the Franka Emika Robot arm with a focus on real-world manipulation and robot learning. Over time, it has proven to be an advantageous tool for other research topics as well as demonstrator for various use cases in the context of robot manipulation. Although it has only been developed and tested for the Franka Emika Robot arm, its code structure is designed in a highly modular way so that it can easily be adapted to other robot platforms by utilizing their respective APIs. MIOS consists of several modules which are coordinated by a core module. An overview of the most important components is shown in Fig. 3.3. The system is capable of various means of communication such as web socket, UDP or RPC. It is also compatible with the widely used ROS framework, enabling the use of a range of different third-party packages. Note that ROS was initially not used as a core component since the modules of MIOS are highly integrated and the node structure of ROS would have brought no benefit. MIOS uses its own mongodb database to save environment data, global parameters and results from learning experiments. Since the code has been documented over the course of this thesis, it is straight-forward to extend in future development to approach not yet covered manipulation problems. Additionally, the software has been prepared to run with as little requirements as possible. Besides the usual requirements of the robot's API *libfranka* itself (e.g. a real-time configured Linux kernel), MIOS has only a few dependencies and can even be used in a Docker environment making it almost completely independent from the host system. It is mostly written in C++ using the 2017 standard, and Python 3.8.

The most important part is the implementation of the GGTWreP model which stretches over several of these modules as explained in Sec. 3.2.4.

### Machine Intelligence Operating System (MIOS)

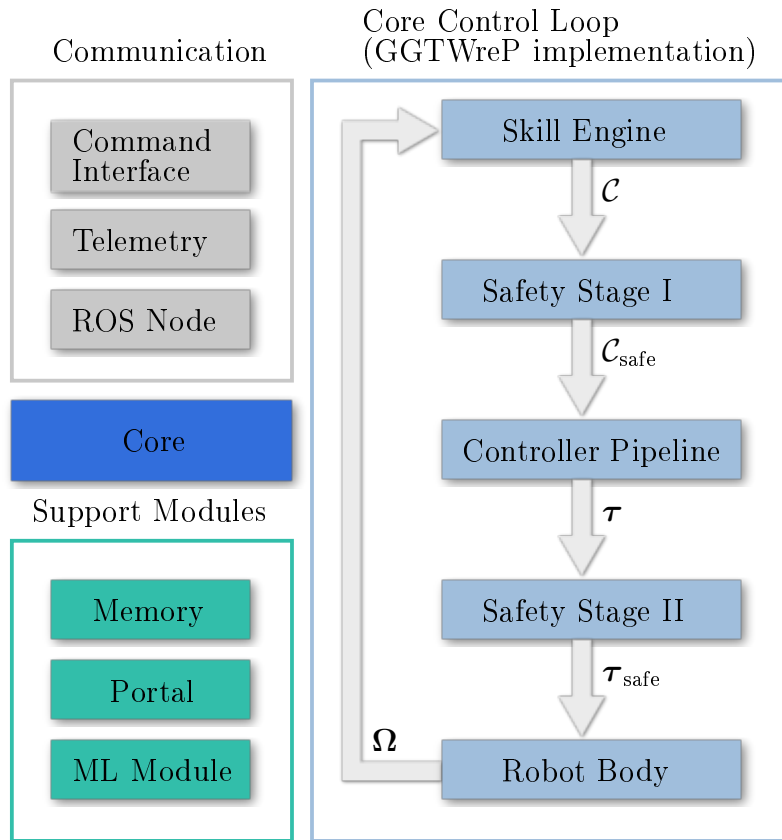


Figure 3.3: Overview of the MIOS modules

### 3.2.2 Design Objectives

The development of MIOS bases on the following design principles:

- *Enablement of Tactile Skills:* The concept of tactile skills must be implemented, i.e. the software must be able to 1) process tactile stimuli from a tactile platform; 2) it must implement a tactile controller; 3) it must provide a platform for tactile policies that command twist-wrench pairs; 4) it must be capable of evaluating skill executions.
- *Skill Optimization:* The software must be capable of using learning algorithms to optimize skills such that no external software packages have to be integrated.
- *Safety:* Safety mechanisms must be built in to ensure a stable and reliable platform for research and demonstrations.
- *Maintainability:* The software must be maintainable, i.e. individual components should be replaceable without introducing breaking changes throughout the system.
- *Realtime:* All control-related parts of the software must be executable in less than 0.5 ms to ensure a reliable real-time process when connected to a robot.



### 3.2.3 Capabilities

MIOS has several controllers at its disposal. Among them are joint and (adaptive) Cartesian Impedance controller, and a force controller. It has two safety stages that ensure that only valid commands are sent to the robot platform. It is capable of running a complete cycle of generating skill and controller commands, sending the commands to the robots and receiving a new robot state within the bounds of a 1 kHz real-time process. A modular toolset of implemented policies enables a programmer to extend the current library of manipulation skills according to the GGTWreP model. It can also directly execute sequences of skills provided for example by the collaborative assembly planner (see Ch. 2.3). A number of communication channels allows peer-to-peer, or even one-to-many communication and telepresence with other robots or user interfaces. MIOS also supports a simplified Python interface for programming complex tasks. Its built-in mongodb database can store information about physical objects and locations and use it to reason about its skills. In summary, MIOS is a platform and an enabler for various research fields ranging from manipulation learning and planning, over motion planning, control and safety, to telepresence and human-robot interaction.

### 3.2.4 Modules

**Core** The core as the central module coordinates all other modules and connects the various components of the command pipeline, i.e. it runs the base control cycle which coordinates the update of the percept struct, queries skill commands and relays them through the controller pipeline to the robot hardware.

**Task Engine** The task engine is the most outer loop in the program logic. It takes care of loading the context of a given task, running and (potentially) recovering it, and event handling such as various error cases. After execution it stores the various data from the result in the short-term memory. A task is a container for any number of skills and can execute additional commands in any manner allowed by the C++ language. When no explicit task is selected to run, an idle task is executed as default.

**Memory** The memory module is responsible for storing any form of data such as task results and environment data. It consists of a short-term memory and a long-term memory. The short-term memory uses only internal RAM to quickly store information and access it. The long-term memory connects to an external mongodb database to permanently store information. The latter is only updated in between task executions.

**Command Interface** The interface module offers a variety of high-level methods for interaction with MIOS via network. It utilizes RPC, websocket and UDP protocols as a framework for json-based communication channels. Additionally, it runs a ROS core, allowing for communication with the ROS ecosystem. The latter can also internally be utilized by skills for publishing or subscribing to ROS topics or make service requests. The methods offered by the various servers are always the same and range from basic hardware commands such as locking/unlocking brakes over information requests to starting/stopping tasks.

**Telemetry** The telemetry module allows a client to subscribe to MIOS and receive telemetry data via UDP with a specified frequency. An already tested use case is a digital twin of the robot using e.g. the UNREAL engine [339].

**Portal** The portal module offers communication services to other modules (e.g. the command interface and the telemetry). These are message-based communication methods as well as continuous streams via UDP.

**Skill Engine** The skill engine takes care of properly loading a skill including its context (any required execution information). It also serves as an interface for the core to execute the skill cycle. At the end of execution the skill engine collects the results from the last skill and adds them to the current task results.

**Controller Pipeline** The controller pipeline is a modular component that offers various control methodologies. In its current implementation it can connect to hardware interfaces based on joint torque, joint velocity, Cartesian twist, joint angles and Cartesian pose. Depending on the interface it makes use of e.g. custom Cartesian impedance and force controllers, nullspace controllers and joint torque controllers. It is easy to extend since a specific pipeline inherits from a base C++ class.

**Safety Stages** MIOS has two safety stages, the first modifies commands on velocity / pose / feedforward force level, the second on joint torque level. The used safety mechanisms are virtual walls, velocity walls, velocity scaling, and limitation of rates and maximum values. Furthermore, built-in checks for invalid values and buffering are implemented.

**Robot Body** The robot body module implements the bridge from MIOS to the used robot hardware. The current implementation is focused on the Franka Emika Robot arm and uses *libfranka*. This module translates the platform-specific robot state into a MIOS-readable percept struct which is then used throughout the other modules. In the other direction it translates the MIOS-specific commands struct into platform-readable commands depending on the used hardware interface.

**Machine Learning Service** The machine learning service is implemented in Python and communicates with the C++ part of MIOS using a websocket client. It runs an engine that calls the interface module in order to execute tasks and receive evaluation information. The engine is used by machine learning services which are based on third-party Python packages. Exemplary implementations are PSP, CMA-ES, Bayesian optimization and various algorithms from the sklearn Python package [340]. The services are instructed by a problem definition class which determines the parameter domain, used cost function, and setup, reset and termination routines for the experiment.

### 3.2.5 GGTWreP Implementation

The GGTWreP model is implemented within MIOS as depicted in Fig. 3.2.1 The semantic and policy layers are mostly covered by three different classes, i.e. skill class,

manipulation primitive class and policy class (abstracted by the skill engine), while the control and hardware layer are taken care of by other modules. In the following all relevant components are briefly described. Additionally, in Alg. 1-6 pseudo code for the most important software routines is given.

**Core Loop** The MIOS core module contains the base loop for skill execution. It calls all necessary components in the right order, see Alg. 1.

---

**Algorithm 1** Core Loop

---

```

1: procedure STARTSKILL(SKILL)
2:   ActiveSkill = skill
3:    $\Omega \leftarrow$  UpdatePercept(Body.GetState())
4:   ActiveSkill.Initialize( $\Omega$ )
5:   ControllerPipeline.Initialize( $\Omega$ )
6: procedure CORELOOP
7:   while  $\mathcal{C}.stop = \text{false}$  do
8:      $\mathcal{C} \leftarrow$  ActiveSkill.SkillCycle( $\Omega$ )
9:      $\tau \leftarrow$  ControllerPipeline.Step( $\Omega, \mathcal{C}$ )
10:     $\Omega \leftarrow$  Body.Control( $\tau$ )
11:     $\Omega \leftarrow$  UpdatePercept( $\Omega$ )

```

---

**Skill Class** The skill class plays a central role for the GGTWreP implementation. It contains the entire semantic layer and the manipulation graph which in turn holds a number of manipulation primitive classes. The skill class is an abstract base class from which the actual skill implementations inherit. The base class has a host of different methods to interface with other modules of MIOS and to provide various support to the concrete skill instantiations. All concrete skills have to implement a set of methods to ensure basic functionality.

The skill class runs its cycle in real time as indicated in Alg. 2. In this cycle the different conditions of the semantic layer are checked and acted upon if triggered. Then, the currently active event conditions are checked and the current primitive is switched if they are triggered. If no conditions were triggered, the currently active manipulation primitive is called to calculate its next command.

**Manipulation Primitive Class** The manipulation primitive class is a container for multiple policies that can be executed in parallel. When queried for a command by the skill class, the primitive in turn queries the policies it contains. It makes sure that the resulting command  $\mathcal{C}$  is valid, such that e.g. no two contradicting pose commands are relayed down the command pipeline. The most important functionality of the manipulation primitive class is shown in Alg. 3.

**Policy Class** The policy class is an abstract base class from which concrete policies can be derived. The concrete policies have to implement basic functionality such as initialization, command calculation and a termination condition. Concrete implemented

---

**Algorithm 2** Skill

---

```

1: procedure INITIALIZE( $\Omega$ )
2:   CycleState  $\leftarrow$  Init
3:   ActiveMP  $\leftarrow$  MPGraph.GetInitialMP( $\Omega$ )
4:   Success  $\leftarrow$  false
5: procedure SKILLCYCLE( $\Omega$ )
6:   if CycleState = Init then
7:     if CheckPreconditions( $\Omega$ ) = true then
8:       CycleState  $\leftarrow$  Execution
9:     else
10:       $\mathcal{C} \leftarrow$  ActiveMP.stop()
11:      CycleState  $\leftarrow$  Execution
12:   if CycleState = Terminate then
13:   if CycleState = Execution then
14:     if CheckErrorConditions( $\Omega$ ) = true then
15:        $\mathcal{C} \leftarrow$  ActiveMP.stop()
16:       CycleState  $\leftarrow$  Terminate
17:     if CheckSuccessConditions( $\Omega$ ) = true then
18:       Success  $\leftarrow$  true
19:     if CheckTerminationConditions( $\Omega$ ) = true and Success = true then
20:        $\mathcal{C} \leftarrow$  ActiveMP.stop()
21:       CycleState  $\leftarrow$  Terminate
22:     if SwitchMP( $\Omega$ ) = true then
23:       ActiveMP  $\leftarrow$  MPGraph.GetNextMP()
24:    $\mathcal{C} \leftarrow$  ActiveMP.GetCommand( $\Omega$ )
   return  $\mathcal{C}$ 

```

---



---

**Algorithm 3** Manipulation Primitive

---

```

1: procedure GETCOMMAND( $\Omega$ )
2:   for p in Policies do  $\mathcal{C} \leftarrow \mathcal{C} +$  p.GetCommand( $\Omega$ )
3:   if CheckCommands( $\mathcal{C}$ ) = false then  $\mathcal{C}.stop =$  true
   return  $\mathcal{C}$ 

```

---

policies are e.g. a point-to-point motion generator, a twist generator, or a Lissajous wrench generator. In Alg. 4 the example of a point-to-point motion generator is shown.

---

**Algorithm 4** Policy
 

---

```

1: procedure INITIALIZE MotionGenerator.Initialize( $T_0, T_g, \dot{\mathbf{X}}_d, \ddot{\mathbf{X}}_d$ )
2: procedure GETCOMMAND( $\Omega$ ) MotionGenerator.Step( $\Omega$ )
3:    $\mathcal{C}.\dot{\mathbf{x}}_d \leftarrow$  MotionGenerator.GetTwist() return  $\mathcal{C}$ 

```

---

**Controller Pipeline Class** The controller pipeline class provides implementations of controllers to bridge the skill commands  $\mathcal{C}$  and the robot hardware. The pipeline class itself is an abstract base class and multiple concrete pipelines can be derived. Concrete pipelines have to take care of proper controller execution and context switches, i.e. the proper blending of commands between skill executions. MIOS in its current state provides different pipelines for the various command interface levels of the Franka Emika Robot arm. In this thesis, a pipeline consisting of a unified force / impedance controller including nullspace control with desired joint torque as output is used, see Alg. 5 for an example.

---

**Algorithm 5** Controller Pipeline
 

---

```

1: procedure INITIALIZE( $\Omega$ )
2:   CartImpCntr.Initialize( $\Omega$ )
3:   NullSpaceCntr.Initialize( $\Omega$ )
4:   ForceCntr.Initialize( $\Omega$ )
5: procedure STEP( $\Omega, \mathcal{C}$ )
6:   CartImpCntr.Step( $\Omega, \mathcal{C}$ )
7:   NullSpaceCntr.Step( $\Omega, \mathcal{C}$ )
8:   ForceCntr.Step( $\Omega, \mathcal{C}$ )
9:    $\boldsymbol{\tau} \leftarrow$  CartImpCntr.Gettorque() + NullSpaceCntr.Gettorque() + ForceC-
      ntr.Gettorque() return  $\boldsymbol{\tau}$ 

```

---

**Robot Body Class** The robot body class is the low-level bridge between MIOS and the FCI of the Franka Emika Robot arm. It uses the API of *libfranka* to establish connection on various command interface levels, e.g. joint torque, Cartesian twist or joint positions. Its main functionality is to connect the specific libfranka control loop with the MIOS core loop, see Alg 6. Furthermore, it provides access to the gripper as well as high-level features such as locking and unlocking the brakes of the robot or searching for its IP address in a local network during the start up phase of MIOS.

---

**Algorithm 6** Panda Body
 

---

```

1: procedure CONTROL( $\boldsymbol{\tau}$ )
2:    $\Omega \leftarrow$  libfranka.Control( $\boldsymbol{\tau}$ ) return  $\Omega$ 

```

---

### 3.2.6 Learning

Manipulation learning is enabled by the learning module which is implemented in Python. The reason for this is the high availability of third-party packages for machine learning in the Python language. The learning module consists of several connected components, as shown in Fig. 3.4.

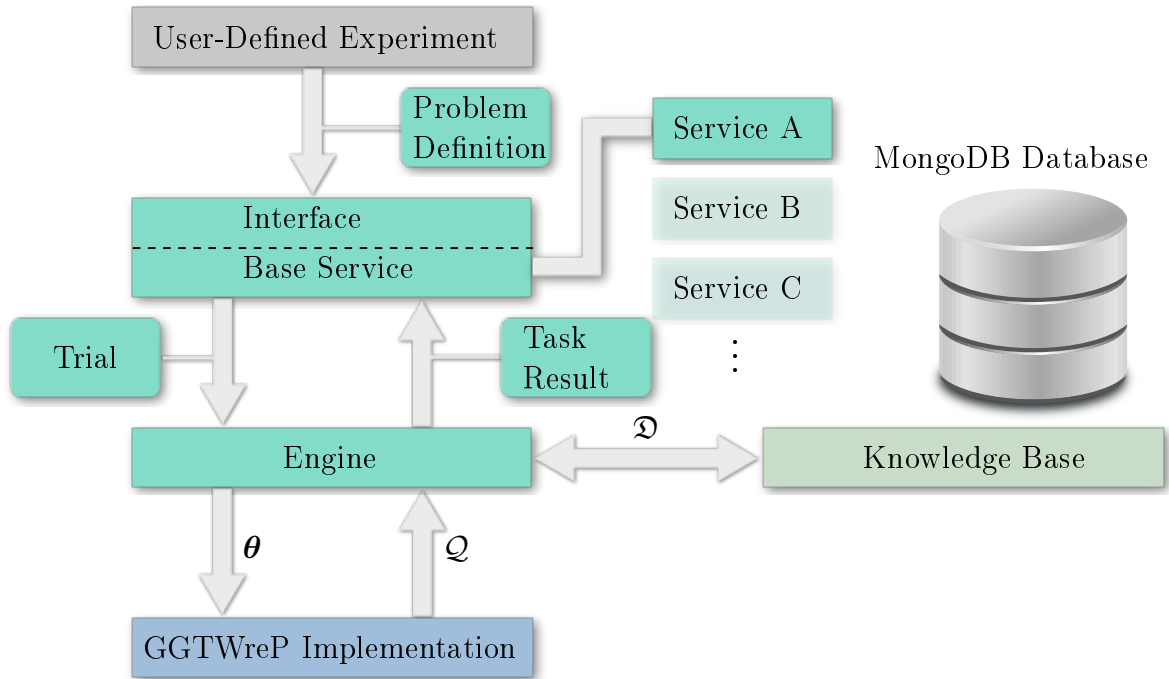


Figure 3.4: Overview and working principle of the learning module

#### 3.2.6.1 Functional Components

**Services** Services implement the actual machine learning methods and are derived from a service base class which provides a common framework and functionality such as initialization of global parameters, parsing of the problem definition and coordination of the learning process. Concrete services have to implement method-specific initialization, learning and termination procedures. Examples for services are the PSP algorithm introduced in Sec. 2.4.1, CMA-ES and Bayesian optimization.

**Interface** The interface is a network-capable point of communication for the learning module attached to the base service class. The communication is currently realized by the RPC protocol and allows for starting and stopping learning experiments remotely.

**Engine** The engine is the core mechanism of the learning module. Services push parameter sets  $\theta$  to the engine's queue which is processed asynchronously by sending the parameters including default execution settings defined in the problem definition to an available agent. Note that one service can work with an arbitrary number of agents to sample a given problem. The engine also takes care of setup, reset and termination routines as well as storing results into a local database.

**Knowledge Base** The knowledge base processes the results that are stored in the database. For example, it can derive a priori knowledge for transfer learning applications.

### 3.2.6.2 Data Structures

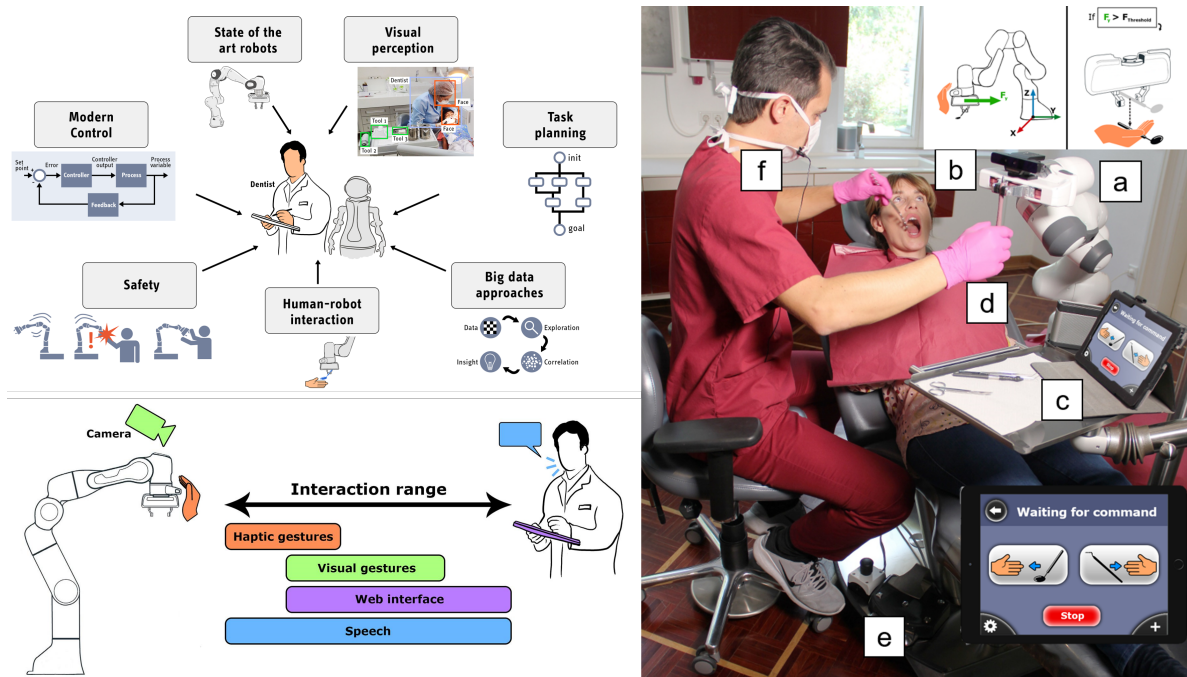
**Problem Definition** The problem definition is a collection of various data items such as the domain  $\mathbb{D}$ , the initial parameters  $\theta_0$  (if applicable to the used ML method) and cost function design (e.g. weights). Furthermore, it calculates the final cost function value according to the metrics in Sec. 2.4.2.

**Task Result** A task result is a simple container that holds all available information from a task execution, e.g. cost function value, heuristics and success indicator.

## 3.3 Validation Experiments: Use Case Integration

This section presents a number of validation experiments in which MIOS has been used as a platform. Each experiment uses, besides the core software, different components, or was even the incentive the extend the software accordingly. It is demonstrated how MIOS and as a consequence the GGTWreP framework and the concept of tactile skills have enabled further research work and publications in various related fields.

### 3.3.1 New Application Domain: Dentronics



**Figure 3.5:** (Top left) The field of dentronics [331]. (Bottom left) The modalities of the devised interaction framework dependent on the communication range [15]. (Right) Experimental prototype setup for the conducted user-study consisting of a Franka Emika Robot arm [5] (a) equipped with an SR300 camera [341] (b), a mobile device (c), colored gloves (d), a pedal (e) and a microphone (f) [15].

In [15] the concept of dentronics was introduced and further developed in [331,342]. This new field is already starting to be recognized by scientists beyond the circle of initial authors [343], and has been featured in IEEE Spectrum [344]. It envisions robotic assistance in typical dental procedures such as assistance during treatment or device disinfection. A set of design principles, namely *human safety*, *human-centered interaction*, *human-robot communication*, *reliable manipulation skills*, and *intuitive to use*, was introduced on which basis a prototypically implemented dentist-robot interaction framework was developed. With the help of a consecutive user-study it could be shown that the concept has the potential to become a realistic application domain for collaborative robots, providing benefit to the dentist. The results also indicate individual user-preferences regarding interaction modalities. In summary, it is hypothesized that a powerful and intuitive multi-modal interaction framework significantly increases the acceptance of robotic assistance in dentistry and is worth to be taken further beyond the first steps.

For the experimental realization a number of software components from MIOS have been used and extended with an interaction framework. The developed framework builds upon the dentronics design principles and combines several communication modalities in order to cover a wide range of dentist-robot distances. These are speech recognition, hand gesture recognition, haptic gestures, a web interface, and a foot pedal. Figure 3.5 shows the setup for the user study.

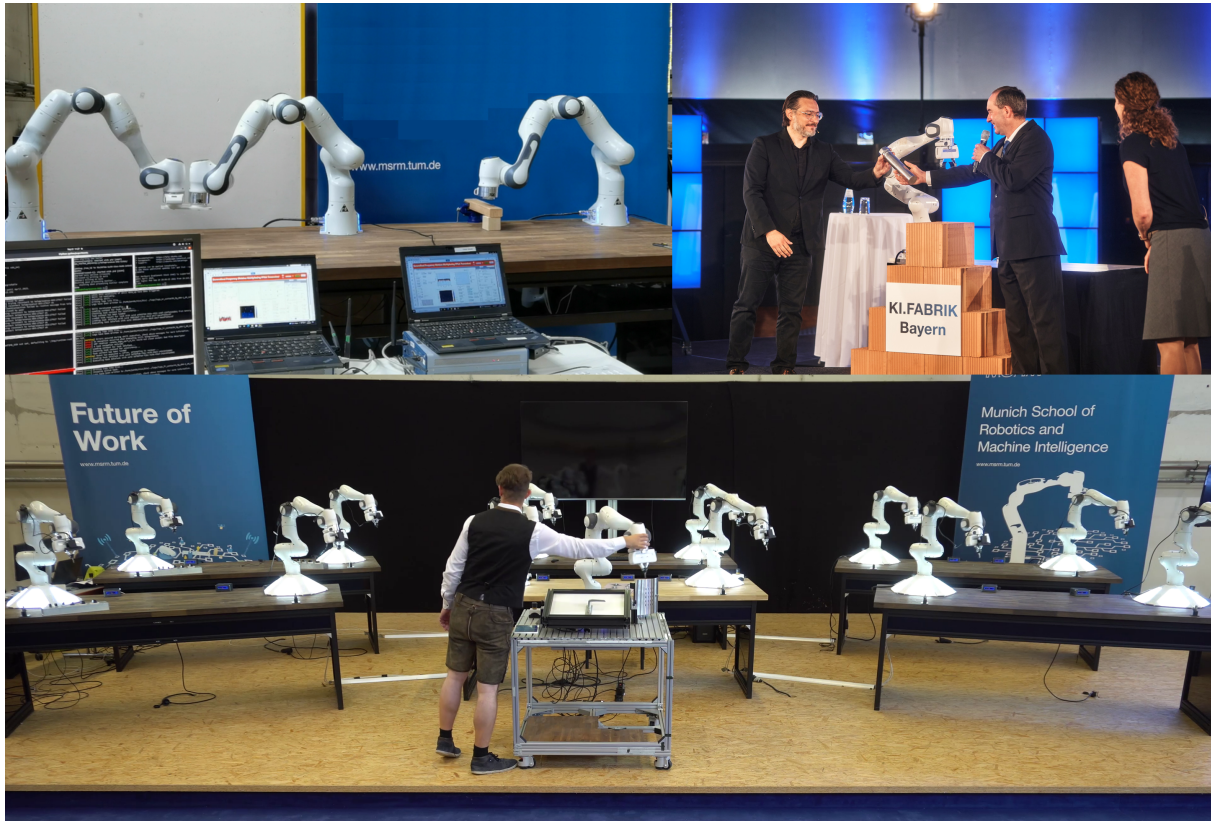
### 3.3.2 Distributed Control: Telepresence

In [332], a pioneering experimental comparison between 5G communication and existing technologies in the context of haptic robotic telepresence was studied. For this purpose, a passivity-based control framework was proposed that is intuitive to understand and to implement as a baseline. For the experiment and evaluation, three 7-DoF tactile robot arms were used as well as a flexible research communication link that implements 5G and 4G LTE as well as commercially available 5G, 4G LTE and WiFi technologies. The telepresence performance of these six communication methods in terms of tracking and force reflection is compared. From the results it is concluded that 5G communication technology is indeed superior in terms of tracking and force reflection, although existing methods may suffice for various, however, basic applications. Therefore it is believed that 5G campus networks will play an important role for wireless robotics applications due to the lower latency and increased stability.

The experimental setup consists of three Franka Emika Robot arms [5] as illustrated in Fig. 3.6 on the top left. The leader robot (LR) and the follower robot (FR) form the teleoperation system, while the third one, the human operator robot (HO), acts the role of a human operator. The motion-controlled HO is rigidly connected to the gravity-compensated LR, such that they both follow the same Cartesian motion. The LR motion passes through the communication layer to the FR. On the other hand, the contact wrench sensed by the FR is sent back to the LR through the same communication layer. Ideally, the FR should follow the exact Cartesian motion of the HO, and the external wrench sensed by the HO should be the same as the sensed external wrench by the FR.

In order to realize this experiment, MIOS was thoroughly extended with a UDP communication interface and telepresence skills. It is possible to use different telepresence modes such as direct joint or Cartesian pose as presented in [332], or a joystick mode, where the leader robot acts as a joystick by employing impedance control. The latter was





**Figure 3.6:** (Top left) The teleoperation setup consists of a leader (LR), a follower (FR) and an human operator robot (HO) which is rigidly connected to the leader for providing repeatable experimental scenarios [332]. (Top right) The kickoff of the KI.Fabrik project © TUM. (Bottom) The one-to-many telepresence showcase in the collective © TUM.

prototypical work for a telemedicine application [345].

Applications based on the telepresence capabilities of MIOS have been shown at numerous occasions. At CeBit 2019 the first 5G-enabled telepresence case for robot arms has been presented together with TU Dresden and Vodafone. It has been used in combination with the collective (see Sec. 3.4.2) to enable external demonstrations. The most impressive example of this was the official opening of the Munich Institute of Robotics and Machine Intelligence (MIRMI) where one robot was used to directly control 36 others at three different sites in Germany. Other noteworthy showcases were the Hannover Messe 2019 where the Chancellor of Germany Angela Merkel shook hands with the Scientific Coordinator of MIRMI across several hundred kilometers using bilateral telepresence, and the official start of the KI.Fabrik project where an operator at the MIRMI labs used a telepresence-controlled robot to insert a time capsule into its holding at the Deutsches Museum, see Fig. 3.6. It has also impacted and directly supported scientific publications such as [346] where a drone has been connected to a Franka Emika Robot arm via telepresence. One showcase that might open up further lines of research is one-to-many telepresence as implemented within the robot collective at MIRMI, see Fig. 3.6 at the bottom.

### 3.3.3 Local Multi-Robot System: Pinakothek der Moderne

In 2021 a complex robotic art installation called *KI.ROBOTIK.DESIGN* was opened in the Pinakothek der Moderne in Munich [347]. It consisted of eight robots equipped with

cameras in front of a slowly moving canvas. Each robot had a number of pens in reach that were used to draw on the canvas. The drawings were images of daily news sites from around the world or ones created by users via an App. A ninth robot used a crank to move the canvas.



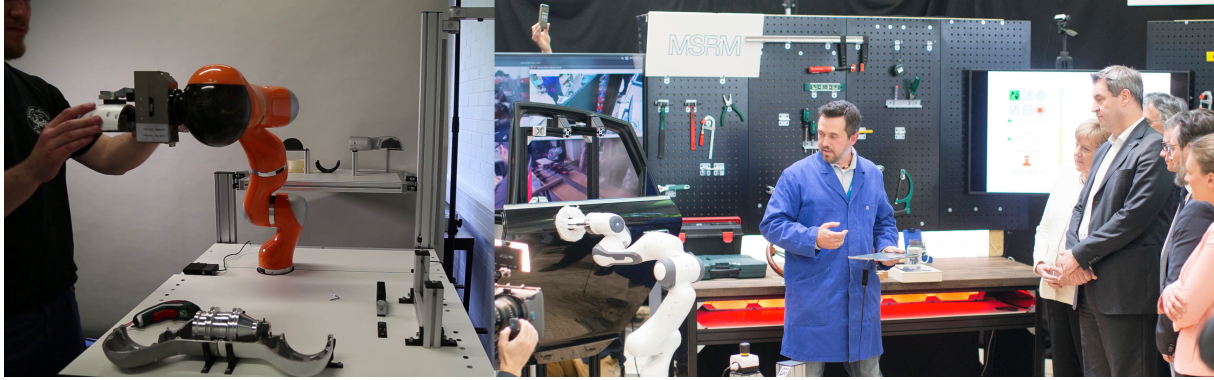
Figure 3.7: KI.ROBOTIC.DESIGN in the Pinakothek der Moderne © TUM

MIOS was utilized to control the robots. During the setup of the exhibition new skills such as draw and crank were added, and its inherent telepresence capabilities have been used. Furthermore, the software stack was extended to handle communication between multiple robots and to provide an interface to container management systems such as Kubernetes. The art installation is depicted in Fig. 3.7.

### 3.3.4 Collaborative Assembly Station

MIOS has been used in a demonstrator that was created based on the collaborative assembly planner presented in [14]. The theoretical foundations and the experimental analysis of this work can be found in Sec. 2.3 and Sec. 4.4, respectively. The demonstrator is shown in Fig. 3.8 on the left.

It was used to showcase several manipulation capabilities such as the compliant interaction with an uncertain environment and safety features such as collision detection. The demonstrator was e.g. featured at the Hannover Messe 2016. Later this was extended to also include a phase-based motion generator that is capable of velocity adaptation depending on human co-workers in the vicinity [348]. This case has also been presented when the German Chancellor Angela Merkel visited MIRMI in 2019, see Fig. 3.8 on the right.



**Figure 3.8:** (Left) The collaborative assembly station [14]. (Right) The later version of the station at the visit of chancellor Angela Merkel © TUM.

## 3.4 Validation Experiments: Learning

This section showcases two demonstration systems that were built to present the learning capabilities of MIOS to the public.

### 3.4.1 Manipulation Learning: Peg-in-Hole

The experimental results of the learning architecture have found their way into a continuously developed demonstrator. It showcased the learning of a difficult peg-in-hole problem and demonstrated the robustness and high performance of the found solution.



**Figure 3.9:** From left to right and top to bottom: A political event at the monastery in Seon © CSU, the opening of the Vodafone 5G lab © Vodafone, Hannover Messe 2019 © TUM, DLD 2019 © TUM, AI Council of Bavaria © TUM.

It used the learning components of MIOS and also served as a testing platform for new developments within the software stack. It was featured on numerous occasions such as the automatica and Hannover Messe trade fairs, various public and political events, and international conferences. Figure 3.9 shows a collage of impressions.

### 3.4.2 Distributed Multi-Robot System: Collective

The collective was the most complex demonstrator coming out of this thesis. To the best of the author's knowledge it is the first long-standing large-scale robot system. At first, its standard configuration consisted of 13 robots, and was later extended to over 70. Each robot was equipped with a camera and its own real-time PC that ran MIOS. It was used for numerous smaller experiments in the fields of manipulation learning, planning, telepresence, human-robot interaction and control. Many of the experiments of this thesis have been done with this setup.



**Figure 3.10:** From left to right and top to bottom: The official opening of the Munich Institute of Robotics and Machine Intelligence (MIRMI) © TUM, the visit of German chancellor Angela Merkel at MIRMI © TUM, the AIBAY conference 2023 © TUM, Falling Walls 2019 © TUM.

The collective was often featured on public and political events, trade shows, and internal demonstrations, most notably the opening ceremony of MIRMI, the visit of the German chancellor Angela Merkel, and the Falling Walls conference. Due to its size and complexity it was only moved once to an external location, namely the official opening of the Munich Institute of Robotics and Machine Intelligence (MIRMI) in the Pinakothek der Moderne. On other occasions, only one robot was moved and connected to the rest via telepresence. Figure 3.10 shows several impressions from various events.

## 3.5 Conclusion

This chapter describes the developed learning architecture that forms the basis for the experiments in Ch. 4. Its main components are the GGTWreP framework, a model for tactile skills, and MIOS, the software stack that implements the GGTWreP framework. Finally, a number of challenging validation experiments underline the unprecedented performance of the developed framework.



# 4

## Experimental Analysis

This chapter discusses the extensive experimental work that verifies the theoretical foundations for tactile skills, skill synthesis, skill learning, and assembly planning. In Sec. 4.1 a verification experiment demonstrates the implementation of 28 different skills that solve a variety of challenging real-world industrial processes. The results indicate a high degree of robustness and performance as well as performance stability. It can also be shown that the tactile policies coming from the skill synthesis process are reusable across different skill classes. In Sec. 4.2 the learning architecture is experimentally verified. First, a number of relevant learning algorithms are compared and evaluated with regard to their performance on a difficult insertion problem, which is considered unsolved in robotics. Second, based on the insights from the first experiments, a novel SVM-based learning algorithm was developed and compared to state-of-the-art deep reinforcement learning approaches on the basis of a key insertion process. The results show superior learning performance, robustness and achieved manipulation performance. Third, in an earlier experiment a transfer effect has been observed that accelerates skill learning when using prior knowledge from an apparently similar skill. In order to investigate this further, a large experimental campaign was conducted that produced several essential insights with regard to similarity among skills such as dependency on geometry and asymmetry in transferability. Then, Sec. 4.3 describes a reference experiment methodology for comparison of human and robot manipulation capabilities. Additionally, the learning performance of the robot was compared with the programming skills of a robot expert. The results indicate that human-level manipulation performance can already be reached for some skills and that autonomous learning at least achieves similar results to a human expert programmer, albeit slower. Finally, in Sec. 4.4 an experiment is shown in which a collaborative assembly planner that is compatible with the developed skill pipeline solves the task allocation problem for a complex assembly, which has also been tested on real-world production cases. This chapter was written based on [14, 23–25, 321, 349]

## 4.1 Taxonomy Verification

In this section the verification experiment for the taxonomy of manipulation skills is presented. First, the general hardware setup and the specific process setups are described. Then the verification process is outlined and the results are shown. Finally, the results are discussed.

### 4.1.1 Experimental Setup

The taxonomy verification experiments are based on a common hardware base setup that consists of the following components.

- A Franka Emika Robot arm [5]: This is a 7-DOF manipulator with link-side joint torque sensors and a 1 kHz torque-level real-time interface, which allows us to directly connect the GGTWreP framework to the system hardware, i.e., the real-time interface FCI [5].
- A Franka Emika Robot hand: A standard two-fingered gripper that is sufficient for a wide range of tasks.
- Intel NUC: A small PC that uses an Intel i7 CPU, 16 GB RAM, and an SSD. <sup>1</sup>
- Software: MIOS is used.

Input processes for the taxonomy of manipulation skills (TMS) are directly derived from established standards such as the German curricula for trainees in metalworking [303], electronics [304], and mechatronics [305]. These standards provide the basis for almost any process in today’s industry by defining boundary conditions, manipulation steps, requirements and objectives. By building on top of standard works the robot manipulation framework is then directly compatible with the current needs of industrial companies. As a first step, the TMS contains processes that range from the domain of machine tending (such as lever operation and button pressing), to assembly (such as insertion), or material processing (such as bending and cutting). To illustrate the power of the framework, 28 real-world manipulation skills were implemented within the GGTWreP framework as described in Table 4.1.

### 4.1.2 Verification Process

For the validation experiment, each skill model was executed 50× on the same setup. A single trial involves executing a particular skill model until it terminates. When appropriate, artificial errors were used to offset the skill’s goal poses in the validation experiment to simulate a more realistic process environment. For example, in typical industrial environments, moving parts of heavy machines cause process disturbances that impact the robot’s precision. The process-specific experiment setups are depicted in Fig. 4.1. Considering the validation experiment, as well as the optimization experiments (both autonomous learning and manual tuning), roughly 6000 trials were run, i.e., executions, for a single skill. Taking into account the optimization times and setup times (i.e., physically adjusting the robot’s environment for the next experiment), the entire experimental work took about one month to complete.

---

<sup>1</sup>The used learning approaches do not require GPU acceleration or distributed computing clusters.





**Table 4.1:** Setup descriptions

Insert	A cylinder ( $e_{x,y,z} \pm 0.003$ m) with 30 mm diameter and very low tolerances ( $< 0.1$ mm), a household key, and an Ethernet plug (both $e_{x,y,z} \pm 0.001$ m).	$\pi_{d,27}$
Tip	A mechanical enter key and two different spring-loaded buttons. For the enter key $\delta_2$ was automatically triggered when the key was hit properly. $e_{x,y,z} \pm 0.001$ m for all three cases.	$\pi_{d,23}$
Drag	A box filled with objects, resulting in roughly 2.1 Kg of weight, was dragged over three different surfaces, i.e., wood, cloth and foil.	$\pi_{d,3}$
Slide	A common computer mouse and three different surfaces, i.e., wood, cloth and foil. The robot had to maintain a contact force of 15 N.	$\pi_{d,6}$
Press Mechanism	A pedal ( $e_{x,y,z} \pm 0.01$ m), a user stop, and a flip switch (both $e_{x,y,z} \pm 0.005$ m). The pedal must be pressed for 1 s, the other two buttons have no minimum press time. The user stop required significantly more force to be pressed down than the other two button variations. The press button skill was only optimized for contact torques, since the execution time is mostly determined by the press time.	$\pi_{d,33}$
Extract	A cylinder ( $e_{x,y,z} \pm 0.003$ m) with 30 mm diameter and very low tolerances ( $< 0.1$ mm), a household key, and an Ethernet plug (both $e_{x,y,z} \pm 0.001$ m).	$\pi_{d,34}$
Cut	A cutter knife on a carton surface, a cloth surface and a foil surface. Success was confirmed manually and by visual inspection by a human experimenter depending on whether the surface has been cut properly.	$\pi_{d,26}$
Grab	An object (HDMI switch in this case) on a table with $e_{x,y,z} \pm 0.005$ m. Note that, the grab skill is very easy to test since it is a position-based task for the most part with little physical interaction. The grab skill was only optimized for execution time due to the minimal amount of physical interaction.	$\pi_{d,35}$
Place	An object (HDMI switch in this case) grasped by the robot to be placed on a flat table with $e_{x,y,z} \pm 0.005$ m. The place skill was only optimized for execution time due to the minimal amount of physical interaction.	$\pi_{d,36}$
Swipe	A stylus on a tablet with $e_{x,y,z} \pm 0.005$ m. The success of the skill was determined visually by the human experimenter depending on a successful swipe operation on the tablet.	$\pi_{d,26}$
Bend	Two wooden plates connected by heavy cables which allow for a reset without too much wear on the setup ( $e_y \pm 0.005$ m).	$\pi_{d,3}$
Turn Mechanism	A key inserted into a lock ( $e_{x,y,z} \pm 0.005$ m).	$\pi_{d,2}$
Slide Off	A battery casing with a slideable lid ( $e_{x,y,z} \pm 0.005$ m). The success of the skill was determined visually by the human experimenter.	$\pi_{d,4}$
Move Mechanism	A common lever ( $e_y \pm 0.005$ m).	$\pi_{d,1}$

relearned) to find the new optimum. Some policies are even directly transferable between different classes. When looking at the building blocks of the selected policies, one might even say that many manipulation processes can be solved by using a small toolset of

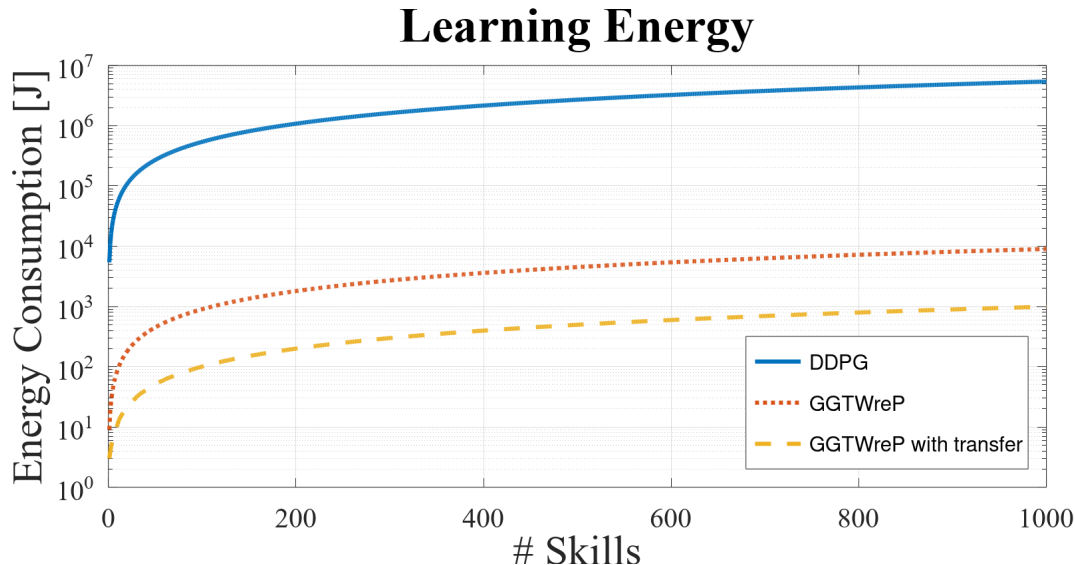
**Table 4.2:** Experimental results for all skills

Skill	Task	Execution Time		Contact Torques	
		Robustness	Value	Robustness	Value
Insertion	Cylinder	94 %	$1.76 \pm 0.36$ s	92 %	$3.21 \pm 0.35$ Nm
	Key	100 %	$1.1 \pm 0.18$ s	98 %	$2.97 \pm 0.244$ Nm
	Ethernet Plug	100 %	$1.04 \pm 0.19$ s	100 %	$2 \pm 0.55$ Nm
Extraction	Cylinder	100 %	$0.35 \pm 0.05$ s	100 %	$6.03 \pm 0.123$ Nm
	Key	100 %	$0.31 \pm 0.02$ s	100 %	$5.46 \pm 1.277$ Nm
	Ethernet Plug	100 %	$0.23 \pm 0.001$ s	100 %	$2.29 \pm 0.023$ Nm
Press Mechanism	Pedal	100 %	N/A	100 %	$4.12 \pm 0.223$ Nm
	Flip Switch	100 %	N/A	100 %	$1.6 \pm 0.083$ Nm
	User Stop	100 %	N/A	98 %	$3.39 \pm 0.17$ Nm
Tip	Enter Key	100 %	$0.82 \pm 0.005$ s	100 %	$0.59 \pm 0.013$ Nm
	Red Button	100 %	$0.69 \pm 0.034$ s	100 %	$1.41 \pm 0.137$ Nm
	White Button	100 %	$0.69 \pm 0.008$ s	100 %	$1.32 \pm 0.023$ Nm
Grab	HDMI Switch	100 %	$2.87 \pm 0.005$ s	N/A	N/A
Place	HDMI Switch	100 %	$2.42 \pm 0.064$ s	N/A	N/A
Slide Object	Wood	100 %	$1.21 \pm 0.008$ s	100 %	$8.34 \pm 0.125$ Nm
	Cloth	100 %	$1.21 \pm 0.005$ s	100 %	$8.69 \pm 0.098$ Nm
	Foil	100 %	$1.21 \pm 0.005$ s	100 %	$9.56 \pm 0.077$ Nm
Drag	Wood	100 %	$1.02 \pm 0.06$ s	100 %	$6.42 \pm 0.04$ Nm
	Cloth	100 %	$1.09 \pm 0.06$ s	100 %	$6.5 \pm 0.016$ Nm
	Foil	100 %	$1.01 \pm 0.008$ s	100 %	$11.86 \pm 0.078$ Nm
Cut	Carton	100 %	$1.69 \pm 0.009$ s	100 %	$7.16 \pm 0.502$ Nm
	Cloth	100 %	$1.75 \pm 0.089$ s	100 %	$6.08 \pm 0.105$ Nm
	Foil	100 %	$1.72 \pm 0.011$ s	100 %	$5.92 \pm 0.145$ Nm
Turn Mechanism	Key	95 %	$0.71 \pm 0.19$ s	100 %	$0.94 \pm 0.345$ Nm
Swipe	Tablet	100 %	$1.4 \pm 0.17$ s	66 %	$3.1 \pm 0.155$ Nm
Move Mechanism	Red Lever	100 %	$1.33 \pm 0.036$ s	86 %	$4 \pm 0.079$ Nm
Bend	Cables	100 %	$1.99 \pm 0.006$ s	100 %	$4.37 \pm 0.195$ Nm
Slide off	Battery Case	98 %	$1.13 \pm 0.66$ s	96 %	$7.22 \pm 0.623$ Nm

building blocks. This result supports the idea that the proposed approach is versatile enough to be relevant for realistic scenarios.

As this approach enables the learning of a wide range of skills in realistic settings, the issue of energy consumption in real-world 24/7 skill acquisition settings becomes a pertinent concern. Therefore, the computational energy required for the presented approach is compared with that of an exemplary state-of-the-art deep learning system, as is illustrated in Figure 4.2. The GGTWreP framework used in this work is compared with the deep deterministic policy gradient method (DDPG) [350]. Furthermore, GGTWreP is shown for the case in which every skill is learned from scratch as well as for the case in which learned solutions can be reused and transferred to subsequent learning. This suggests that using current state-of-the-art data-based methods to learn many skills may require significant resource demands, as was anticipated, for example, in [351]. However, using the GGTWreP framework requires an order of magnitude lower energy, and even significantly less energy than that with transfer learning.

For the deep deterministic policy gradient (DDPG) framework, convergence was observed



**Figure 4.2:** A comparison of the required energy to learn a great number of skills. The deep deterministic policy gradient (DDPG) algorithm is compared with the GGTWreP framework (see Sec. 3.2.5 both with and without transfer learning).

after  $\sim 300$  trials. However, contrary to previous expectations, no robust solution was achieved most of the time. A trial includes executing a skill and calculating the learning algorithm. Learning a skill with the use of prior knowledge from another skill within the same subclass (level-1 transfer) reduces the number of required trials to  $\sim 30$  for GGTWreP, and to  $\sim 60$  for transfer across subclasses (level-2 transfer). DDPG, in its current form, does not achieve a reliable transfer. To compare all calculations with the same widely available computing platform, an Intel NUC with an i7 processor was used that has an average power consumption of 30 W. To compare the power consumption for the different frameworks, only the CPU time allocated for the learning algorithm was considered, which is 1 ms for GGTWreP and 0.6 s for DDPG. For a conservative analysis, it is assumed that the CPU worked at full capacity in both approaches. Consequently, the energy consumption per trial is estimated to be 0.03 J for GGTWreP and 18 J for DDPG. The overall energy consumption is

$$e_{n_s} = \sum_{i=0}^{n_s} \mathbf{d}_i, \quad (4.1)$$

where  $e_{n_s}$  denotes the energy consumption for  $n_s$  number of sequentially learned skills, and  $\mathbf{d}_i$  is the vector for the required number of trials for each learned skill. To make a reasonable comparison with transfer learning, it is assumed that the goal is to sequentially learn 100 skill subclasses with 10 instances for each of a single skill class.

## 4.2 Tactile Skill Learning

### 4.2.1 Comparative Analysis of Algorithms for Skill Learning

This experiment provides a proof-of-concept for the learning architecture introduced in Ch. 3 and its theoretical foundations in Ch. 2. Four different learning algorithms are compared and the results discussed.

### 4.2.1.1 Experimental Setup

The experimental setup consists of a Franka Emika Panda robot [102] and three variations of the insertion problem as shown in Fig. 4.3, i.e. a key, a puzzle piece and an aluminum cylinder.



**Figure 4.3:** Experimental setup, puzzle (top right), key (bottom left) and cylinder (bottom right)

The three tasks are as follows:

- **Puzzle:** The puzzle piece is an equilateral triangle with a side length of 0.075 m. The hole has a depth of 0.005 m. The tolerances between puzzle piece and hole are  $< 0.1$  mm and there are no chamfers. Note that the same basic setup as in [352] was used.
- **Key:** The lock has a depth of  $d = 0.0023$  m. The lock and the key have chamfers to make the initial insertion easier by design.
- **Cylinder:** The aluminum cylinder has a diameter of 0.02 m and the hole a depth 0.035 m. The tolerances are  $\ll 0.1$  mm and there is a 0.5 mm chamfer on the cylinder. The hole has no walls which results in a higher chance of getting stuck during insertion, which further increases the difficulty.

The experiment has the following routine:

1. The learning problems are given as described above.
2. The insertion skill is selected from the taxonomy (note that the parameters and structure of the GGTWreP model in this early experiment differ from the current version. Please refer to [23] for details.).

3. The objects (key, puzzle and cylinder) are taught kinesthetically by a human expert.
4. Latin hypercube sampling (LHS), particle swarm optimization (PSO), Bayesian optimization (BO) and covariance matrix adaptation evolution strategy (CMA-ES) are selected as learning algorithms.
5. Each combination of insertion problem and algorithm was run ten times to ensure a statistical confidence.

For Bayesian optimization (BO) the spearmint software package [313,314,353,354] was used in combination with predictive entropy search with constraints (PESC) [316] as acquisition function. For evolutionary algorithms Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [317,318] was selected and for Particle Swarm Optimization (PSO) the standard implementation [355]. Furthermore, Latin Hypercube Sampling (LHS) [319,356] is utilized, an uninformed grid-based sampler. The learning algorithms are configured as follows.

- LHS: The parameter space is sampled at 75 points.
- CMA-ES: The algorithm ran for 15 generations with a population of 5 individuals and an initial standard deviation of  $\sigma_0 = 0.1$ . These values were found by initial try-outs. The initial centroid was set in the middle of the parameter space.
- PSO: 25 particles were used and the algorithm ran for 3 episodes. The acceleration constants were set to  $c_1 = 2$  and  $c_2 = 2$  (default values).
- BO: The algorithm is initialized with 5 equally distributed random samples from the parameter space.

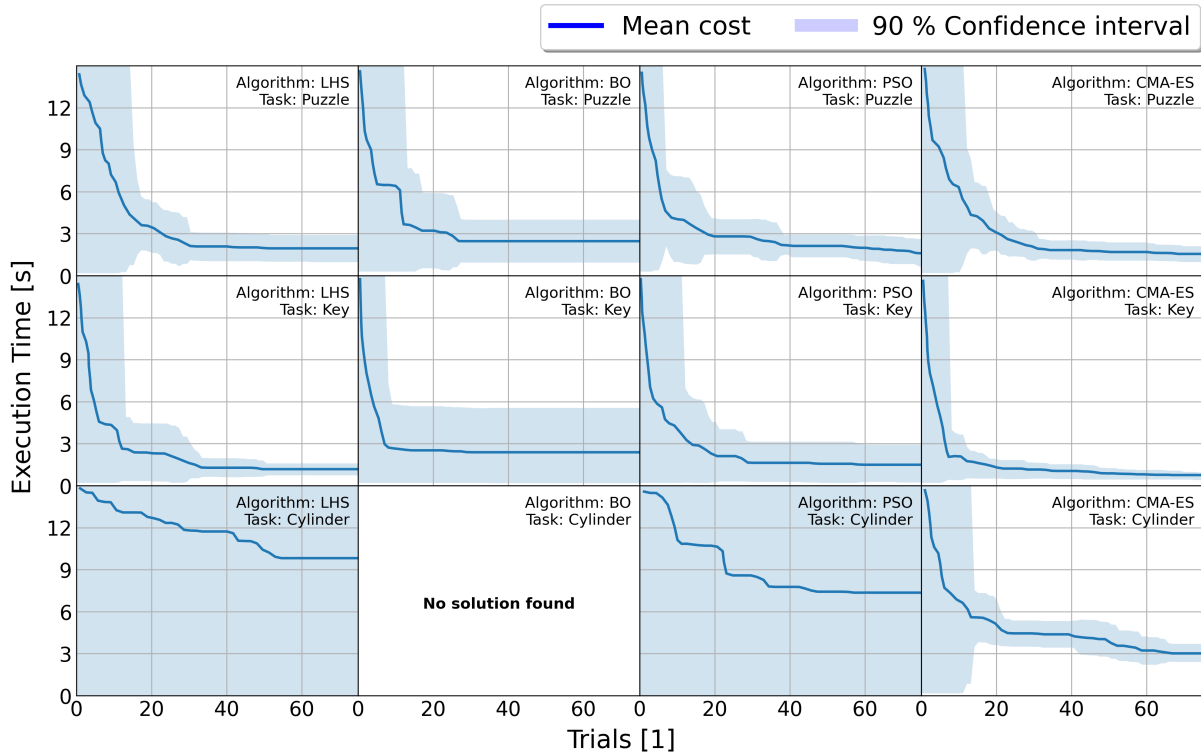
Execution time was used as cost function  $\mathcal{Q}_c$ . The maximum skill execution time was set to  $t_{\max} = 15$  seconds. The heuristic  $\mathcal{Q}_h$  of the insertion skill is given as the distance to its goal pose defined by the *Container* object of the insertion skill.

#### 4.2.1.2 Results

The performance results are shown in Fig. 4.4. The blue line indicates the execution time over trials averaged over all ten experiments per problem/algorithm combination. The grey area indicates the 90% confidence interval. The results of BO for learning to insert the cylinder are not included since it was not able to find a solution in the given time frame.

These results show that all four algorithms are suited to a certain degree to learn easier variations of the insertion task such as the puzzle and the key. However, it has to be noted that Bayesian optimization on average finds solutions not as good as the other methods. Furthermore, the confidence interval is notably larger. It also terminates early into the experiment since the model was at some point not able to find further suitable candidates. This might indicate a solution space with high noise and discontinuities that is difficult to model.

The comparison with LHS indicates that the GGTWreP approach sufficiently reduces the complexity of the manipulation skill design. Finding solutions with practically relevant



**Figure 4.4:** Experimental results. The columns correspond to the learning algorithms (LHS, BO, PSO, CMA-ES) and the rows to the tasks (Puzzle, Key, Cylinder). The blue line denotes the cost of an algorithm/task combination averaged over 10 experiments. The light blue area denotes the 90 % confidence interval.

execution times by sampling rather than explicit learning is possible for at least lower tactile difficulty.

The plot showing the LHS results for the cylinder insertion indicates the high complexity of the problem. Random sampling leads to feasible solutions, however, the confidence interval is too large to conclude assurance. PSO achieves better solutions, yet it also has a very low confidence. CMA-ES outperforms both methods and is able to find a solution that is better in terms of absolute cost as well as confidence.

Considering the best performing algorithm CMA-ES, a feasible solution for any of the tasks was already found after 2 – 4 minutes and optimized after 5 – 20 minutes depending on the task, significantly outperforming existing approaches for learning insertion. Note also that with the exception of BO no noteworthy computation time was necessary.

## 4.2.2 Comparison with Deep Reinforcement Learning

This experiment compares the parameter space partition (PSP) algorithm introduced in [321] in combination with the GGTWreP framework to state-of-the-art deep learning methods for a difficult insertion problem.

### 4.2.2.1 Experimental Setup

The setup consists of a Franka Emika Robot arm and a key insertion task, see Fig. 4.5. The lock has a depth of  $d = 0.0023$  m. The lock and the key have chamfers to make the initial insertion easier by design.



**Figure 4.5:** Experiment setup consisting of a robot arm and a key/lock combination

The experiment was setup as follows:

1. The problem task is given by the key/lock setup.
2. The insertion skill from the manipulation skill taxonomy was used.
3. The problem was taught kinesthetically by a human expert.
4. Multiple algorithms were compared, i.e. hierarchical relative entropy policy search (HiREPS), covariance matrix adaptation evolution strategy (CMA-ES), parameter space partition (PSP) and four deep reinforcement learning approaches, namely deep deterministic policy gradient (DDPG), asynchronous advantage actor-critic (A3C), twin delayed deep deterministic policy gradient (TD3) and soft actor critic (SAC).
5. Ten experiments for each algorithm were run, each consisting of 10 episodes and 20 trials per episode.

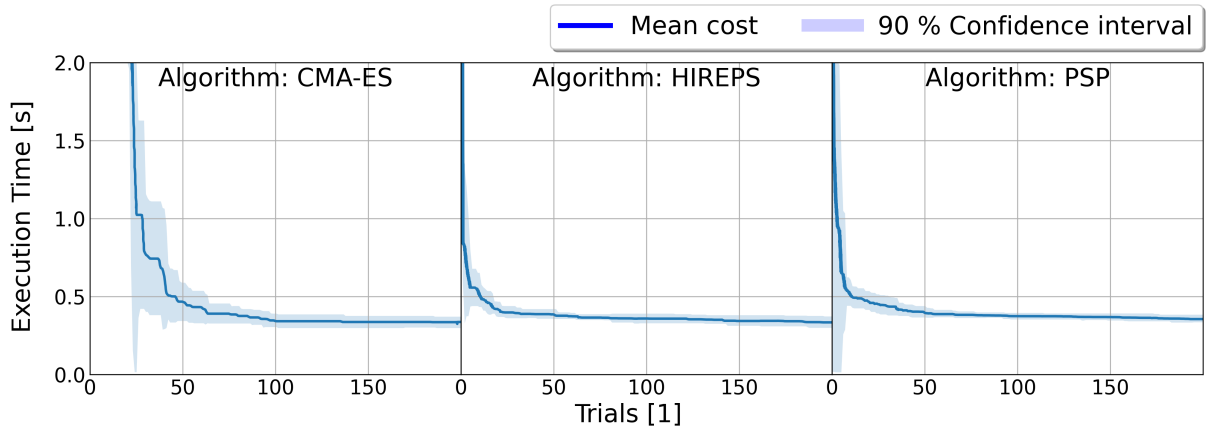
The neural network-based approaches are applied to the insertion phase only, while approach and extraction are handled by GGTWreP-based skills. HiREPS, CMA-ES and PSP make use of the GGTWreP framework. The neural network-based methods are configured with a state space of 21 dimensions, i.e.  $\mathbf{q}$ ,  $\dot{\mathbf{q}}$  and  $\boldsymbol{\tau}_{\text{ext}}$  and an action space consisting of the 7 dimensional motor torques  $\boldsymbol{\tau}_d$ .

#### 4.2.2.2 Results

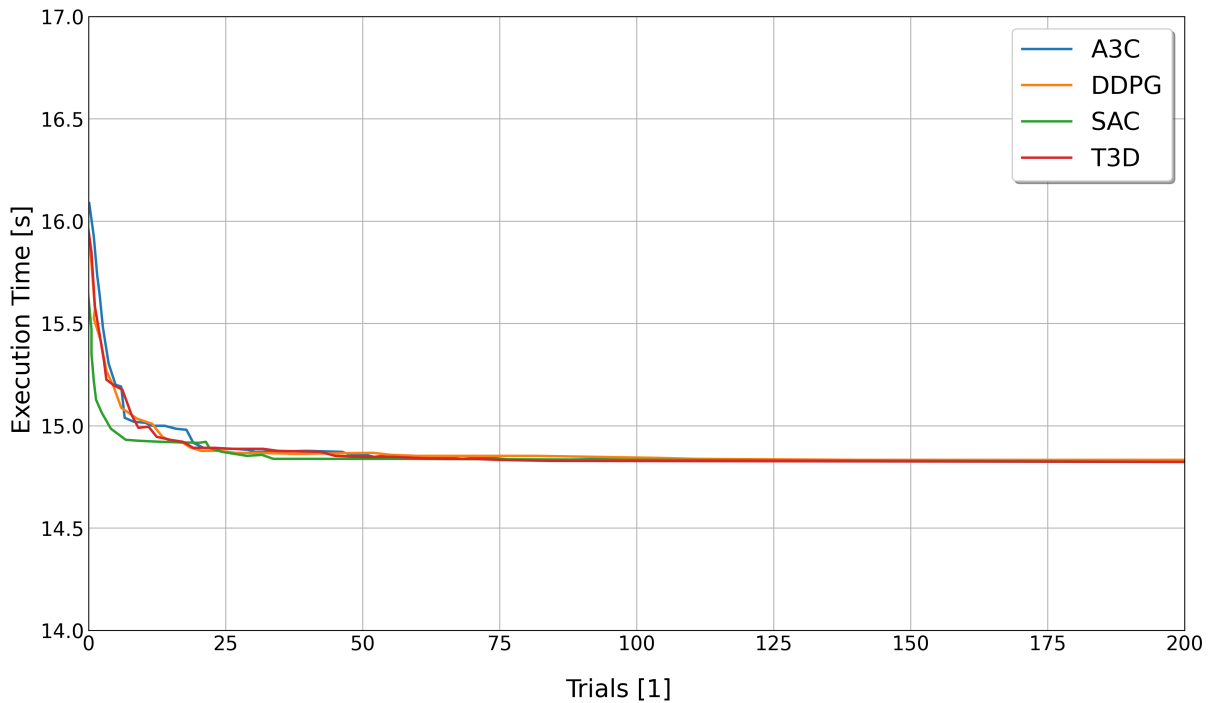
Figures 4.6 and 4.7, and Tab. 4.3 show the results for the applied algorithms. None of the neural network-based learners was able to solve the task in the experiments. There is a clear learning behavior for PSP and HiREPS which seem to be similar. CMA-ES, while also reaching a similar optimized cost, shows a distinctively slower learning rate. Table 4.3 shows the average optimized cost over 10 experiments for each algorithm.

Both PSP and HiREPS seem to fare similar in the experiment. However, the variance of HiREPS is higher, while the average cost is slightly lower. This stems from the fact that HiREPS finds unreliable solutions, which can yield slightly lower costs but are prone to failure. The mean and variance of the parameters for those costs were calculated and the data for parameter tuples within this range were checked. It could be verified that for those parameter tuples the probability of yielding those superior costs was around 38 %, while 62 % resulted in far higher costs (due to very high noise) or failure. The same





**Figure 4.6:** Results of the comparison experiment for HiREPS, CMA-ES, and PSP. The blue line denotes the cost averaged over 10 experiments. The light blue area denotes the 90 % confidence interval.



**Figure 4.7:** Results of the comparison experiment for A3C, DDPG, SAC, and T3D

**Table 4.3:** Optimized costs for successful trials

	PSP	HiREPS	CMAES	DDPG	TD3	SAC	A3C
Result	success	success	success	failure	failure	failure	failure
$\bar{c}_o$	0.3514	0.3345	0.3384	14.842	14.816	14.834	14.831
$\text{var}(\bar{c}_o)$	0.0225	0.0366	0.0377	0.136	0.193	0.163	0.118

investigation applied to PSP did not result in such findings. Tab. 4.4 shows the average percentage of successful trials per episode over the course of learning.

From the results one can see that the PSP algorithm is able to discriminate between successful and unsuccessful parameter spaces and therefore finds reliable optimized solu-

Episode	1	2	3	4	5	6	7	8	9	10
PSP	50.7	71.2	85.5	89.8	91.2	94.2	96.2	98.5	98.3	98.8
HiREPS	57.5	54.5	60.5	61.5	69.5	67.5	72.0	71.5	69.0	69.5
CMAES	2.8	21.1	43.3	59.4	71.1	65.0	71.1	68.9	71.7	69.2

**Table 4.4:** Successful trials over episodes in % (averaged over experiments)

tions. Also, due to its nature, it is able to not only provide a singular solution, but rather a solution subspace within the initial search space. CMA-ES needs several episodes to achieve a distinctive percentage of successful proposals, but still only manages to get on par with HiREPS.

Thus, in this experiment the suitability of the PSP algorithm for difficult real-world robot learning in combination with the GGTWreP framework was demonstrated. Not only are the converged solutions robust, but also the ones sampled directly from the learner, which allows to improve while achieving success to a certain margin. The general idea of defining and iteratively reducing sub spaces of interest seems a promising direction and allows to optimize parameters and to approximate sub spaces of parameters which may be applicable to a specific optimization problem variation. Furthermore, the algorithm in combination with the GGTWreP framework was compared to the state of the art in robotic manipulation learning which is not able to find solutions to the considered (very challenging) manipulation problem, at least within the given frame of 200 trials.

These findings suggest that employing a meaningful structure may be beneficial for learning very challenging real-world manipulation problems such as key insertion.

### 4.2.3 Skill Transfer Learning

This section presents experimental results on transfer learning. The experiments follow up on the discovery of a transfer effect as described in Sec. 2.4.3. First, the experimental performance of state-of-the-art deep reinforcement learning to solve the transfer problem in tactile skill learning<sup>2</sup> is analyzed. Specifically, the well-adopted gold-standard algorithms soft-actor-critic (SAC) [357] and deep deterministic policy gradient (DDPG) [358] were evaluated. Then a large-scale experimental campaign to investigate the transfer learning capabilities of the MIOS framework is described. The results are examined with regard to the hypothesis made in Sec. 2.4.3.

#### 4.2.3.1 Experimental Setup

The experimental setup consists of nine different, albeit somewhat similar insertion problems. The objects to insert are six cylinders with diameters ranging from 10 to 60 mm and three different keys (see Fig. 4.8). The clearance for the cylinders is  $< 0.1$  mm. Therefore, it is on an industrial level; thus, it is a non-trivial challenge that is still not considered solved in the manipulation and control community. Furthermore, to make the task even more challenging, the holes have no walls. This increases the chance of jamming with the cylinders during insertion - an effect that cannot yet be simulated. The keys are

<sup>2</sup>Note that the focus is explicitly on blind, tactile-only manipulation, i.e., no cameras or other external sensors were considered.



**Figure 4.8:** The analyzed reference tasks consist of six cylinders with increasing diameters and three different common household keys (bottom left). Five robots were used in total for the experiment.

common household keys with a typical clearance of  $< 1$  mm, and their particular insertion mechanics are also far from being replicable in physics simulators today. In accordance with the taxonomy hierarchy there are level-0 transfers (each task is paired with itself), level-1 transfers (any cylinder is paired with any other cylinder, or any key with any other key), and level-2 transfers (any cylinder is paired with any key) (see Sec. 2.4.3).

Five Franka Emika Robot arms [5] were used to learn the skills in parallel where possible. Each robot was connected to an off-the-shelf PC with an Intel i7 processor, 8 GB RAM, an SSD, and no additional graphics card. All calculations by the learning algorithm were done on the PCs locally using MIOS (see Sec. 3.2).

#### 4.2.3.2 Experimental Procedure

A robot learning a skill is referred to as an experiment, and one experiment consists of a number of trials. A trial consists of the following procedure.

1. firmly grasp the object with two-finger gripper
2. move to a kinesthetically taught approach pose
3. execute the insertion skill until it is successful, or until an error occurs
4. reset the skill by extracting the object and retract back to approach pose

The movement and extraction skills used to prepare and reset every trial were not part of the learning process but they were implemented by the GGTWreP framework and were parameterized beforehand.

**Deep Reinforcement Learning** In the deep RL experiment, only the cylinders with the sizes 30, 40, and 60 mm were considered. They were learned independently, and then the obtained policy was transferred to the other two cylinder scenarios, respectively. For each cylinder an experiment that consisted of 500 trials was run. For statistical confidence, 10 experiments per cylinder were performed, so there were 15.000 trials in total. The deep RL methods were applied to the insertion problem in two different configurations: 1) The same initial pose at each trial and 2) with offsets in the  $(x - y)$ -plane of up to  $\pm 5$  mm (random uniform distribution). The reward function

$$r(t) = 100(-\delta z(t) + 3|\delta x(t)| + 3|\delta y(t)|), \quad (4.2)$$

was used, where  $\delta z$  is the change in height. Therefore, the reward correlates linearly with the achieved insertion depth at each time step.  $|\delta x(t)|$  and  $|\delta y(t)|$  are the changes in the deviation from the the goal position. When the item is successfully inserted, a reward of  $r_{\text{succ}} = 20$  is added. The network architecture is deep deterministic policy gradient (DDPG) [358], and the implementation was kept close to the originally proposed design. The parameters of the implementation can be found in Tab. 4.5. The soft-actor-critic (SAC) algorithm did not perform sufficiently well, therefore, its parameterization is not shown for sake of brevity.

**Table 4.5:** DDPG parameters

Parameter	Value
Optimizer	Adam [359]
Learning rate (critic and actor)	$3 * 10^{-4}$
Discount( $\gamma$ )	0.99
Replay buffer size	$10^6$
Number of hidden layers	3
Number of units per hidden layer	256
Number of samples per mini-batch	64
Activation function	ReLU
Output activation function	TanH
Target smoothing coefficient ( $\tau$ )	0.005
Action dimension size	6
Observation dimension size	18
Initial exploration phase w/o learning (observations)	1000

Actions are limited to  $\pm 1$  and scaled by factor  $\mathbf{s}_a = [5, 5, 10, 1, 1, 2]$  prior to robot application. The factor  $\mathbf{s}_a$  was found empirically during experimentation.

Since no successful insertions were observed for pure end-to-end learning on a joint level, a 6-dimensional wrench (action  $\in \mathbb{R}^6$ ) in Cartesian space was applied. For the sensory feedback, the Cartesian pose, velocity, and external end effector wrench (each  $\in \mathbb{R}^6$ ) was selected. Therefore, the stacked observation vector is  $\mathbb{R}^{18}$ . Each element of the action was limited between  $\pm 1N$ . Before applying the action to the robot, it was multiplied by the factor  $\mathbf{s}_a = [5, 5, 10, 1, 1, 2]$ . These values were found experimentally, and they seem to define a rather confined box, where the resulting forces are still high enough to solve the insertion task but also low enough to not trigger excessive forces during unwanted collisions and contact. Furthermore, when the values are too high, the robot tends to get stuck in undirected exploration phases, presumably because the task region of interest

(ROI) is rather small. There is also the possibility of building up to much energy, which leads to safety violations during impact. Finally, the amount of movement upward was limited to 1 cm above the insertion hole, as subsequent downwards movement exerts too much kinetic energy during impact.

**GGTWreP** In the GGTWreP experiment, every skill was learned by itself without any prior knowledge, and the results were saved to a database. Then, every skill was learned again with the prior knowledge of every other skill so that every combination of skills was covered. They were also learned again using their own prior knowledge. In addition, every experiment was repeated 10 times to ensure statistical confidence. The CMA-ES algorithm (see Sec. 2.4.1) was used to learn the skills, specifically, the implementation available at [360]. The number of generations was set to 10. The other parameters were set according to the recommendations in [360], which resulted in a total number of 130 trials per experiment. Note that slightly better solutions may be found after more than 130 trials since real-world manipulation learning is usually very noisy. Learning with or without prior knowledge may also end at slightly different costs after 130 trials, but it should eventually converge to the same minimum for  $t \rightarrow \infty$ . For learning skills without prior knowledge,  $c_0$  was set to  $0.2 \theta_{i,\max}$  for every parameter  $i$ , except for  $\Delta \mathbf{x}$  (an intentional deviation from the goal pose), which was set to  $0.5 \theta_{i,\max}$ . This choice reflects an arbitrary, but safe, initialization. It considers that the process starts with low velocities and contact forces. The total number of experiments amounts to 900, and the total number of evaluations, and thus real-world interactions, to 117.000. The experiment took about five entire days to complete.

#### 4.2.3.3 Knowledge Transfer: DDPG

The known network parameters  $\theta_{\text{nn},i}^*$  from the source task  $\iota_i$  are used and applied to the target task  $\iota_j$ ,  $\theta_{\text{nn},j} = \theta_{\text{nn},i}^*$ . This is straightforward, as all the networks in the deep RL implementation have the same number of layers and nodes. Note that no similarities between the network parameters  $\theta_{\text{nn}}$  were examined in any of the tasks, and no form of soft update was used for this transfer. After this transfer, learning was not continued, as the networks did not converge reliably. It is acknowledged that this transfer approach is extremely greedy; however, it is not feasible to generate any kind of weighted mean due to the nonlinear nature of deep neural networks. In addition, the need to find generally applicable reordering methods for such networks in order to make them comparable, and the time needed to approximate their potential value (i.e., weight), are out of the scope of this work.

#### 4.2.3.4 Knowledge Transfer: GGTWreP

To transfer knowledge from task  $\iota_i$  to task  $\iota_j$  within the GGTWreP framework, a set of already found optimal parameters  $\theta^*$  from  $\iota_i$  is used as an initial centroid  $c_0$  for  $\iota_j$  (i.e.,  $c_{0,j} = \theta_i^*$ ). This method of transferring knowledge is intentionally used, since, to the best of the author's knowledge, no general methods have been established yet for such new types of transfer learning. In order to acknowledge the simplicity of the solution, it is referred to this transfer learning as a "greedy type," as it is entirely uninformed about optimal prior choices. The optimal parameter set  $\theta_i^*$  for  $\iota_i$  is extracted from the successful

trials of the experiment. However, simply using the parameters from the trial that generated the lowest cost may not be sufficiently robust. A set of parameters is considered robust if similar parameter sets lead to similar costs  $\mathcal{Q}$ , while strong fluctuations indicate lack of robustness. Therefore, successful trials are clustered first. From the parameters for the best performing cluster, a weighted mean is calculated as

$$\boldsymbol{\theta}^* = \omega_1 \boldsymbol{\theta}_1 + \omega_2 \boldsymbol{\theta}_2 + \dots + \omega_n \boldsymbol{\theta}_n, \quad (4.3)$$

where  $n$  is the number of parameter sets in the best performing cluster. The associated weights  $\omega_1, \dots, \omega_n$  decrease in logarithmic fashion and are normalized.

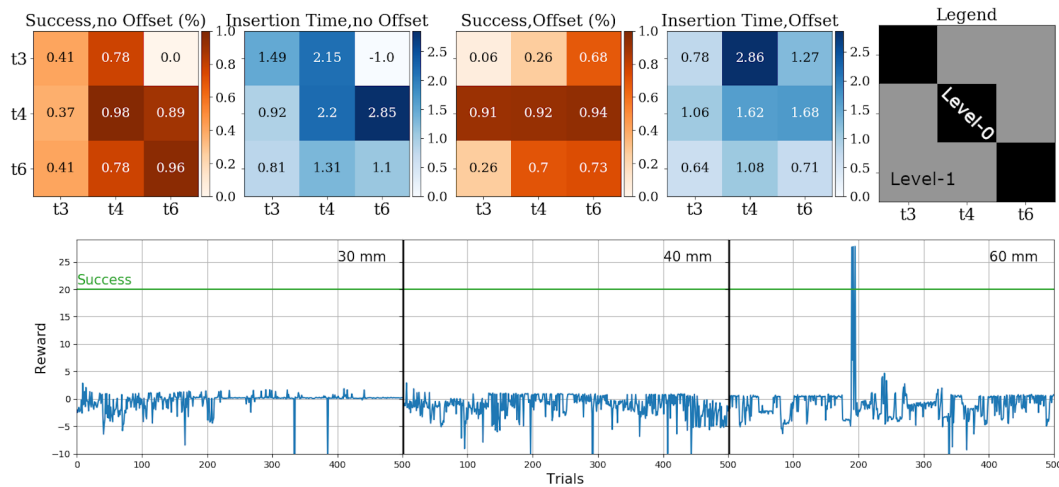
$$\omega_i = \frac{\ln(n + 0.5) - \ln(i)}{\sum_{j=1}^n \ln(n + 0.5) - \ln(j)} \quad (4.4)$$

For task  $\iota_i$ , the weighted mean of the successful parameter sets  $\boldsymbol{\theta}_i^*$  is used.

#### 4.2.3.5 Results: Deep Reinforcement Learning

The results of the deep reinforcement learning experiments are depicted in Fig. 4.9. No convergence could be observed for the soft-actor-critic (SAC) algorithm, despite best efforts, as is shown in the time plots at the bottom of Fig. 4.9. The time plots for the deep deterministic policy gradient (DDPG) algorithm [358] were omitted, since learning the tasks worked in principle and the transfer learning results showcased by the matrices at the top of Fig. 4.9 were the focus of the experiment. The insertion times were considered as cost for the Level-0 and Level-1 knowledge transfers. The columns of the matrices denote the source tasks and the rows present the target tasks. During learning, the network parameters  $\boldsymbol{\theta}$  were saved every 100 trials, and were examined thereafter regarding reliable task execution. Note that this experiment was split into two parts as mentioned above. In the first experiment, there was no initial offset between initial pose and hole, while a random offset for the initial pose was applied in the second experiment. The first and second matrix show the results without an offset, and the third and fourth matrix show the results with uniform random offsets of up to  $\pm 5$  mm.

The time plots (bottom of 4.9) for the SAC algorithm show representative examples of the learning behavior. The  $y$ -axis shows the reward that was obtained for each trail, and the  $x$ -axis shows the trials. One exemplary episode is shown for each of the pegs. No solution could be obtained for any peg with this algorithm; therefore, no knowledge transfer could be observed. For the pegs with a diameter of 30 mm or 40 mm, not one successful insertion could be observed. The small rewards reflect the algorithm's struggle to manage even slight insertions. Successful insertions for the 60 mm peg were observed during learning, however, when the successful parameters were tested without any additional learning or exploration, the performance was not reliable. In total, 25 different parameter sets per peg were applied and there were 50 trials per set over the course of learning. Note that since the observed level-0 transfer in the DDPG experiment without an offset was superior to that with an offset, any experiments with an initial offset were omitted for the SAC experiment. Furthermore, the experimental setup was the same for DDPG and SAC. This result was in stark contrast to initial expectations, as SAC performed best in simulations [361]. One can only hypothesize why this discrepancy occurred. Presumably, the extremely complex nature of the examined contact-rich tasks differs dramatically from



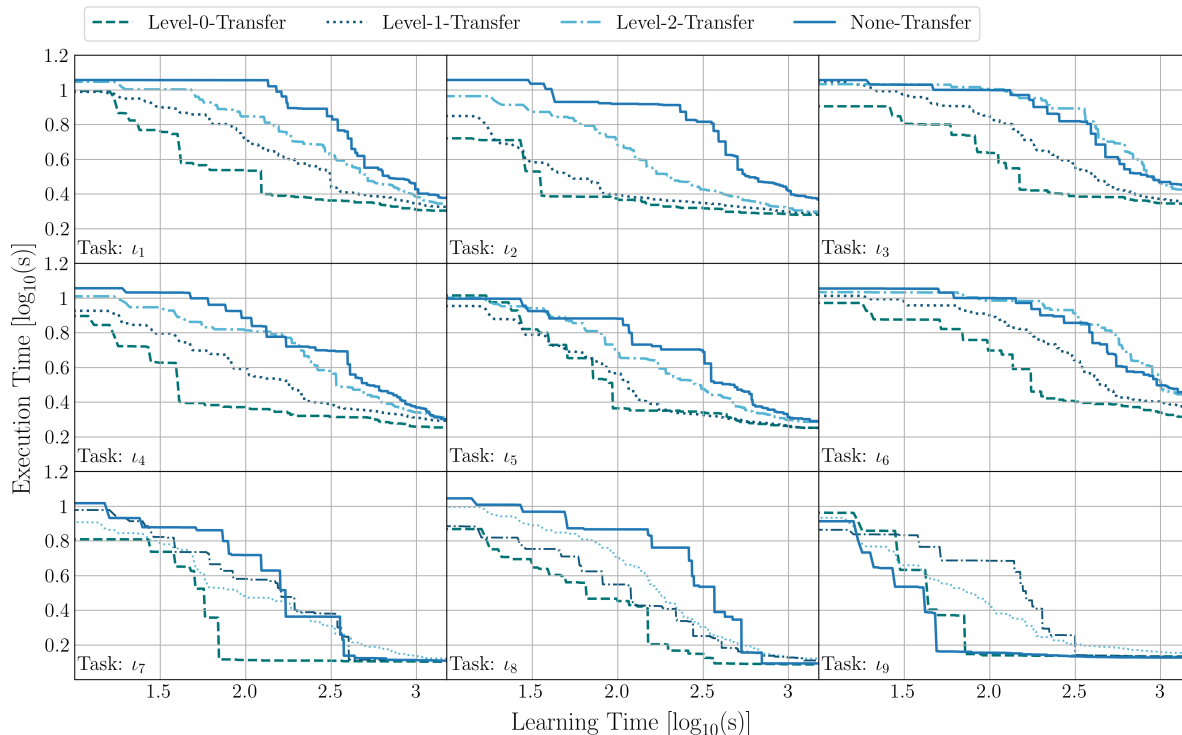
**Figure 4.9: Top)** The transfer learning performance of DDPG. **Bottom)** A representation of the SAC algorithm’s learning performance in the insertion task. Graphs are shown for the 30 mm, 40 mm, and 60 mm pegs. The rewards above the green line indicate successful trials. For the 30 mm and the 40 mm pegs, no successful trials were observed. While successful trials were observed for the 60 mm peg, these were random and not systemically exploited.

standard simulation problems and is indeed very hard to accurately capture and simulate physically.

**Observations** The DDPG algorithm did not converge reliably in the experiment. Reasonable policies were learned sometimes, with success rates of  $> 90\%$  in the first 100 trials of an experiment. Regardless, the network later got stuck in policies with success rates of only  $< 2\%$ . In the experiment, vanilla end-to-end learning without carefully integrated model knowledge (e.g., by suitable coordinate choice and cost function design) was not able to successfully insert a cylinder, as was first indicated in [321]. Moreover, an explicit reward function needs to be formulated to incorporate additional knowledge, including a substantial part of the insertion policy itself. Ultimately, knowledge generation and transfer were unreliable, although a slight knowledge transfer was observable. Apparently, no systematic representation of learned tasks, such as a dependable insertion policy, could be transferred.

#### 4.2.3.6 Results: GGTWreP-based Learning

The results of the large-scale GGTWreP experiment are depicted in Figures 4.10-4.12. From observing the respective speedup results, it can be concluded that using prior knowledge (in this case, in the form of initial policy and control parameters) in real-world tactile manipulation learning is beneficial even to very simple transfer mechanisms and may lead to a significant increase in the learning speed. Interestingly, the learning speed decreased in some cases, which hints at possible adverse effects when attempting to use prior knowledge. Figure 4.11 depicts a similar behavior for the learning success rate of skills. Specifically, based on the results, the following observations can be made.

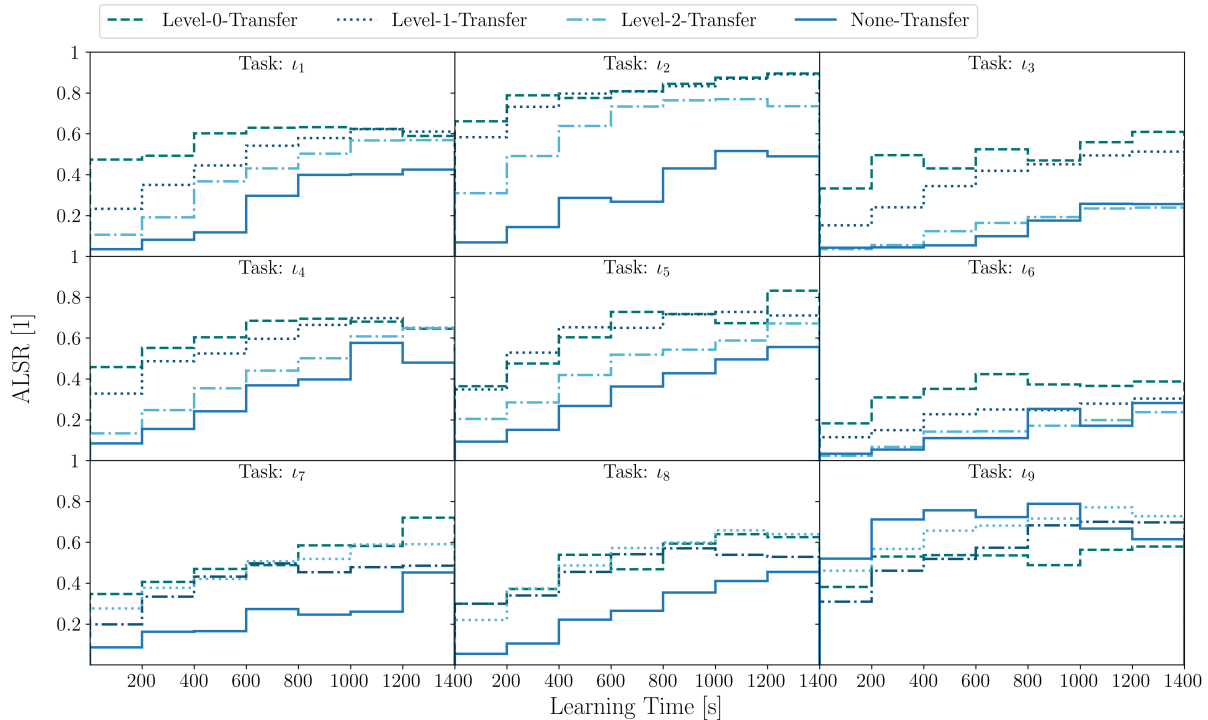


**Figure 4.10:** The logarithmic cost evolution of learning a single skill without prior knowledge (solid blue lines) and the mean cost when using a level-0 (dashed lines), level-1 (dotted lines), and level-2 transfer (dash-dotted lines), respectively. Each task was learned using prior knowledge from every other task. For clarity, the results from each subclass (cylinders and keys) were grouped by averaging the resulting costs and plotting the mean. The cylinder insertion tasks are  $t_1$  to  $t_6$ , and the key insertion tasks are  $t_7$  to  $t_9$ .

**Observations** In addition to the aforementioned benefits of transfer learning to the learning performance in general, the asymmetry of the empirical transferability (ET), learning effort ratio (LER), and speedup stands out when inspecting the results. Thus, it can be concluded that transferring knowledge from task  $t_i$  to task  $t_j$  may significantly speed up the learning process, but, in turn, this is not necessarily the case when knowledge is transferred from task  $t_j$  to task  $t_i$ .

Investigating the experimental results more deeply, it was observed that the learning time for all cylinders was significantly lower when using prior knowledge from other cylinders, while the same holds for the keys with prior knowledge from other keys (i.e., level-0 and level-1 transfers). On the other hand, when using prior knowledge from one of the keys on the cylinders or the other way around (i.e., level-2 transfer), the effect was considerably smaller - sometimes negligible or, in a few cases, even negative. This indicates that the transfer learning performance depends on the specific geometry of a given task. Although, no obvious geometric property stands out such as a diameter change (e.g. in level-1 transfers between the cylinders). In the context of the experiment, one can only speculate about the reasons for this. However, the most influential factors appear to be friction and almost unnoticeable tolerance differences between the cylinder-hole combinations. Note that the data differences in the transfer learning performance among the defined transfer levels support the taxonomy hierarchy proposed in Sec. 2.1.3. In direct support, the initial ALSR also indicates higher initial success rates when prior knowledge was used. Tasks such as the third key ( $t_9$ ) are the exception, since using prior knowledge to learn it had a mostly negative effect (e.g., a slowdown of 50 %) even when using its own knowledge.





**Figure 4.11:** Average learning success rate (ALSR) in the same pattern as Figure 4.10. For better readability, the ALSR was approximated by time-window averaging and plotted as stair function. The black dashed line is the theoretical optimal case with a slope of 1. For each skill, the average ALSR is shown when taking prior knowledge from the cylinder tasks and from the key. Note that an  $ALSR < 1$  means that not enough learning time has passed to fully converge to a feasible solution. However, the difference between raw learning and transfer learning over time is of interest.

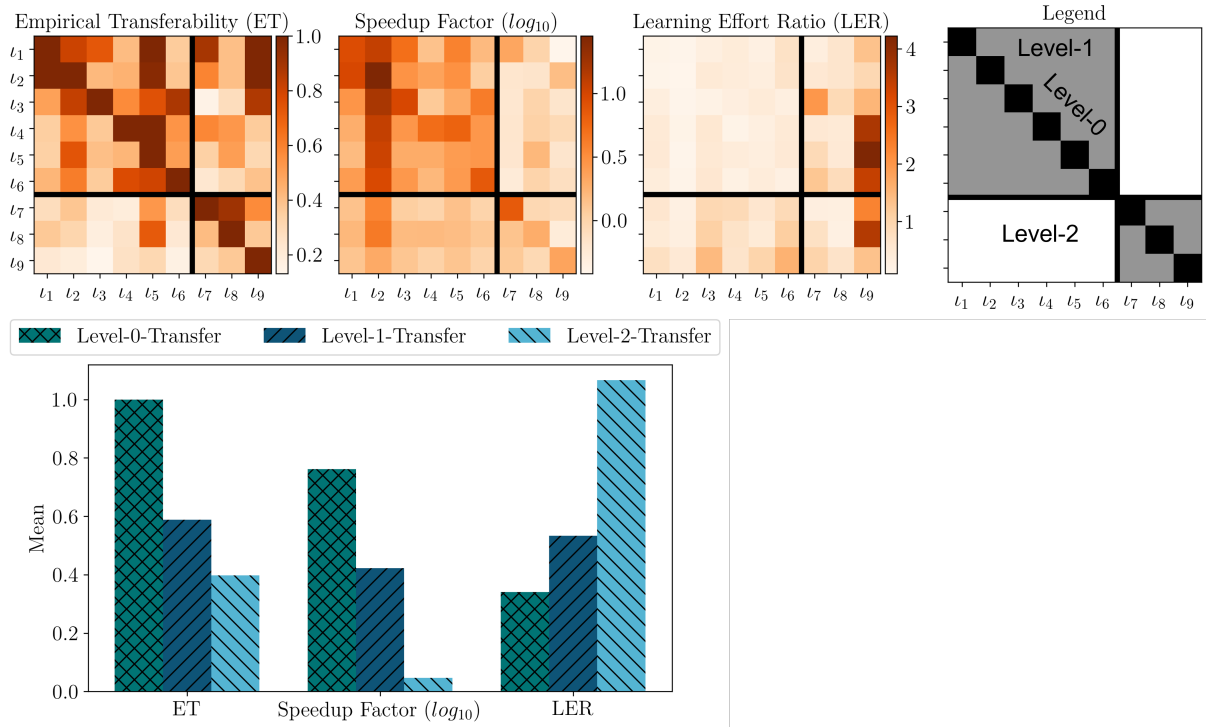
This is presumably because task  $t_9$  belonged to the special case of simple-to-learn tasks, as is indicated by the high ALSR when learning  $t_9$  from scratch. This particular key has a simple geometry and can be inserted into the lock at a wide range of angles. Also, it requires very little pushing force and works even with significant initial offsets. This presumably leads to fewer learning benefits from previous knowledge, as it is already quite simple to learn the skill without any prior knowledge.

### 4.3 Performance Comparison: Robot vs. Human

In this section an experimental case study is presented which compares (learned) robot manipulation performance to human capabilities. The study consists of a tactile-only (meaning no vision) manipulation task, composed of a set of challenging tactile skills. The performance of the robot is compared against that of an adult human in terms of both execution time as well as learning time, see Fig. 4.13. In this the capabilities of the GGTWreP framework for manipulation skill learning are further demonstrated.

#### 4.3.1 Task Description

For the comparative case study the following single-handed task was investigated. Given is a board with a lock and a button on it, and a matching key for the lock is nearby in a holding slit. A visual depiction of the task is shown in Fig. 4.14. The task has the following steps.

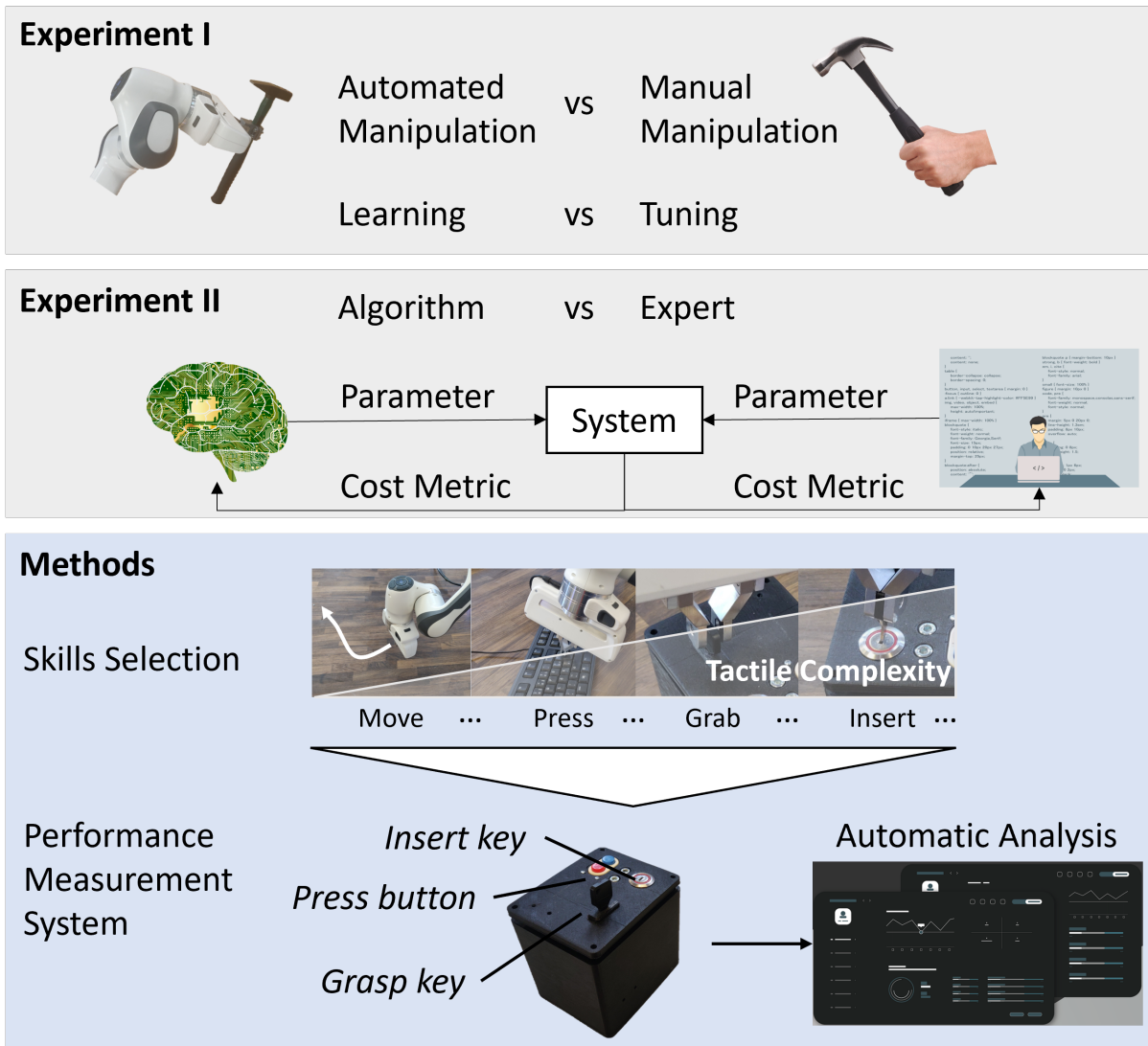


**Figure 4.12: Top)** The ET, speedup factor, and LER for each task combination. The black dividers separate the level-0, level-1, and level-2 transfers (see the legend on the right). On the left side, the target tasks are denoted, and, at the bottom, the source tasks. The color maps are depicted to the right of each plot. **Bottom)** The mean LER, ET, and speedup factor ( $\log_{10}$ ) over all tasks for the respective transfer levels.

1. Grab the key from its storage.
2. Insert the key into the lock.
3. Turn the key by 90 degrees clockwise and turn it back.
4. Extract the key from the lock.
5. Place the key back into its storage.
6. Press the button.

Each of these steps can be achieved by a single skill. These particular skills were chosen to cover multiple important aspects of manipulation. These are handling objects (*grab* and *place*), difficult contact-rich manipulation with changing contact state (*insertion* and *extraction*), manipulation without changing contact state (turn) and fast manipulation (*press button*). Additionally, phases of free motion connect the skills. The skills are implemented using the GGTWreP framework (see Sec. 3.1).

The experiment compares 1) manual programming of the robot and autonomous learning to achieve optimal performance, and 2) the achieved performance with that of a human adult in terms of execution time. Note that in this case study the focus lies on execution time as a means of comparing human and robot performance since it is 1) the most interesting one and 2) the simplest to implement. Other cost functions such as energy or contact forces would require significant effort on the measuring part for the human and are therefore out of scope.



**Figure 4.13:** The intention of the case study is to draw a comparison between robot and human manipulation performance. This also entails a comparison between the learning and tuning approaches towards this performance level. For the presented case study a number of skills of varying tactile complexity was selected and a reference task was constructed based on them.

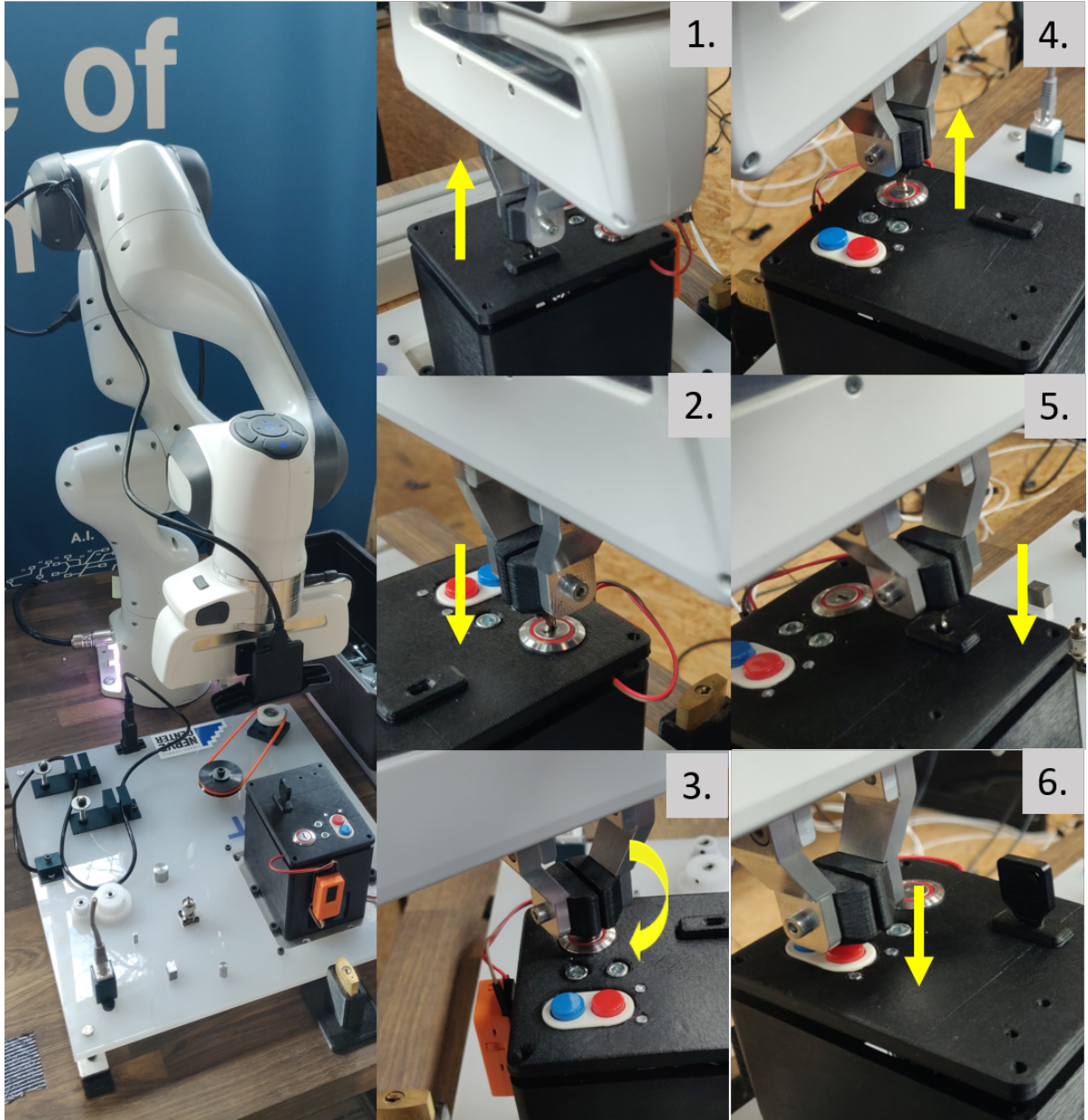
### 4.3.1.1 Expert Tuning

The manual parameter tuning was done by a robot expert skill by skill with execution time as an optimization goal in mind. The tuned parameters per skill and their domains from which valid values were chosen are given in Tab. 4.6. To measure the execution time, each skill as well as the overall task was executed ten times and the average was taken. In addition, another expert robot programmer not familiar with the GGTWreP framework or MIOS was asked to tune the parameters of the skills while the time they required to achieve comparable performance was measured.

### 4.3.1.2 Parameter Learning

The parameter learning was done for each skill separately as follows.

1. The seven learning problems are given by the study task as described in Sec. 4.3.1. The seventh problem is the basic move skill responsible for the free-motion phases.



**Figure 4.14:** The left picture shows the overall setup. The right side shows the respective steps of the task. The arrows indicate the direction of movement for the task steps: 1. Grab the key, 2. Insert the key, 3. Turn the key and turn in back, 4. Extract the key, 5. Place the key, 6. Press the button.

2. The appropriate skills were selected from the taxonomy.
3. All skills were grounded by kinesthetic teaching.
4. The CMA-ES algorithm was used to learn solutions for the skills. The meta-parameters were chosen according to the recommendations found in [360] and the number of generations was set to  $n_g = 20$ . The initial centroid  $c_0$  was set to  $0.1\theta_{i,\max}$  for every parameter  $i$  for all skills, except for  $\Delta\mathbf{x}$  of the insertion skill which was set to  $0.5\theta_{i,\max}$ . This choice reflects a safe initialization considering the process starts with low velocities and contact forces.
5. Each of the skills was learned separately and each experiment was repeated ten

**Table 4.6:** Parameter domains

Parameter	Dimension	Domain
Move		
$\dot{\mathbf{X}}_d$	$3 \times 1$	$([0, 0.5], [0, 1], \text{N/A})$
$\ddot{\mathbf{X}}_d$	$3 \times 1$	$([0, 1], [0, 4], \text{N/A})$
Grab		
$\dot{\mathbf{X}}_{d,g}$	$3 \times 1$	$([0, 0.5], [0, 1], [0, 2])$
$\ddot{\mathbf{X}}_{d,g}$	$3 \times 1$	$([0, 1], [0, 4], \text{N/A})$
Insertion		
$\dot{\mathbf{X}}_{d,i}$	$3 \times 1$	$([0, 0.5], [0, 1], \text{N/A})$
$\ddot{\mathbf{X}}_{d,i}$	$3 \times 1$	$([0, 1], [0, 4], \text{N/A})$
$\mathbf{a}$	$6 \times 1$	$([0, 10], [0, 10], [0, 10], [0, 3], [0, 3], \text{N/A})$
$\mathbf{f}$	$6 \times 1$	$([0, 2], [0, 2], [0, 2], [0, 2], [0, 2], \text{N/A})$
$\Delta \mathbf{X}$	$6 \times 1$	$([-0.01, 0.01], [-0.01, 0.01], \text{N/A},$ $[-0.0175, 0.0175], [-0.0175, 0.0175], \text{N/A})$
$\dot{x}_{\min}$	1	$([0, 0.5])$
Turn		
$\dot{\mathbf{X}}_{d,t}$	$3 \times 1$	$(\text{N/A}, [0, 2.5], \text{N/A})$
$\ddot{\mathbf{X}}_{d,t}$	$3 \times 1$	$(\text{N/A}, [0, 25], \text{N/A})$
Extraction		
$\dot{\mathbf{X}}_{d,e}$	$3 \times 1$	$([0, 0.5], [0, 1], \text{N/A})$
$\ddot{\mathbf{X}}_{d,e}$	$3 \times 1$	$([0, 1], [0, 4], \text{N/A})$
$\mathbf{a}$	$6 \times 1$	$([0, 10], [0, 10], [0, 10], [0, 3], [0, 3], \text{N/A})$
$\mathbf{f}$	$6 \times 1$	$([0, 2], [0, 2], [0, 2], [0, 2], [0, 2], \text{N/A})$
$\dot{x}_{\min}$	1	$([0, 0.5])$
Place		
$\dot{\mathbf{X}}_{d,p}$	$3 \times 1$	$([0, 0.5], [0, 1], [0, 2])$
$\ddot{\mathbf{X}}_{d,p}$	$3 \times 1$	$([0, 1], [0, 4], \text{N/A})$
Press Button		
$\dot{\mathbf{X}}_{d,p}$	$3 \times 1$	$([0, 0.5], [0, 1], \text{N/A})$
$\ddot{\mathbf{X}}_{d,p}$	$3 \times 1$	$([0, 1], [0, 4], \text{N/A})$
$f_{\text{push}}$	1	$([0, 10])$
Controller		
$K$	$6 \times 1$	$([0, 2000], [0, 2000], [0, 2000], [0, 200], [0, 200], [0, 200])$

times resulting in 70 learning experiments in total. Each skill was optimized for execution time with a maximum of  $t_{\max} = 5$  s.

Note that the approach velocities and accelerations were not learned for any of the skills

but instead a separate move skill was learned. This skill is not explicitly part of the experiment task but its results are used for the approach phases of the various skills.

### 4.3.1.3 Measuring Human Performance

The human performance was measured in terms of execution time per skill and per overall task. The study protocol was as follows:

1. The general setup and the task are explained to the participant. The task is demonstrated once.
2. Each skill is demonstrated once.
3. The participant trained each skill until they were able to reliably execute it.
4. Each skill was performed three times with normal speed and three times fast with distinct pauses in between and the performance was recorded per video.
5. The participant performed the overall task at subjective normal speed. The performance was recorded per video.
6. The participant performed the overall task as fast as they could. The performance was recorded per video.

From the recorded videos the execution times per skill and overall task were extracted by always taking the time from the first movement until the hand stopped again. The task was performed by five healthy human participants between the age of 25 – 35 on the same setup as the robot. Note that the human participants could rely on their eyesight whereas the robot was blind i.e. no cameras were involved. However, the robot knew the exact poses of all objects and locations.

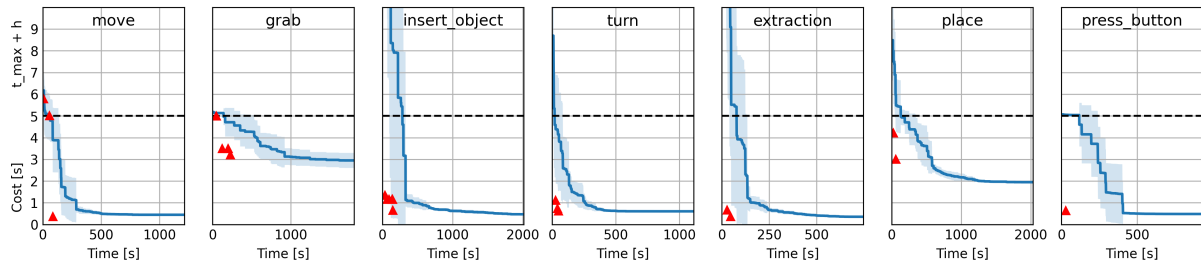
## 4.3.2 Case Study

### 4.3.2.1 Experimental Setup

The experimental setup looks as follows. A Franka Emika Robot arm [362] is mounted on a table and has a board with the described task before it. Initially, the key is placed in the storage slit. The buttons on the board are connected to a micro controller which sends a conformation to the robot when a button has been successfully pressed. Figure 4.14 shows the overall setup and the single task steps. Note that the used two-fingered gripper is one of the most used gripper designs in real-world robotics and is therefore representative in such a case study. The overall task as well as the single steps in terms of execution time are compared.

### 4.3.2.2 Results

**Task Learning** Figure 4.15 shows the learning process for each of the skills as well as the progress of the expert programmer. The graphs show the cost  $Q_c$  over learning time averaged over  $n_l = 10$  experiments. The light blue area around it denotes the 95% confidence interval. The dashed line separates the learning guided by the heuristic  $t_{\max} + h$



**Figure 4.15:** Learning process for all skills. The blue line denotes the cost over time averaged over 10 experiments. The light blue area denotes the 95% confidence interval. The red triangles denote the tuning steps of the expert programmer.

and learning guided by the optimization criterion  $t_e$ . The red triangles denote the tuning steps of the expert. Note that the skills were tuned in the order of *move*, *grab*, *place*, *insertion*, *extraction*, *turn*, *press button*.

The robot is able to reliably learn and optimize each of the skills within a few minutes. This demonstrates the applicability of the GGTWreP framework to a wider range of manipulation learning problems. The total time to learn and sufficiently optimize all skills autonomously was roughly about 80 minutes on average whereas the expert programmer took only about 12 minutes.

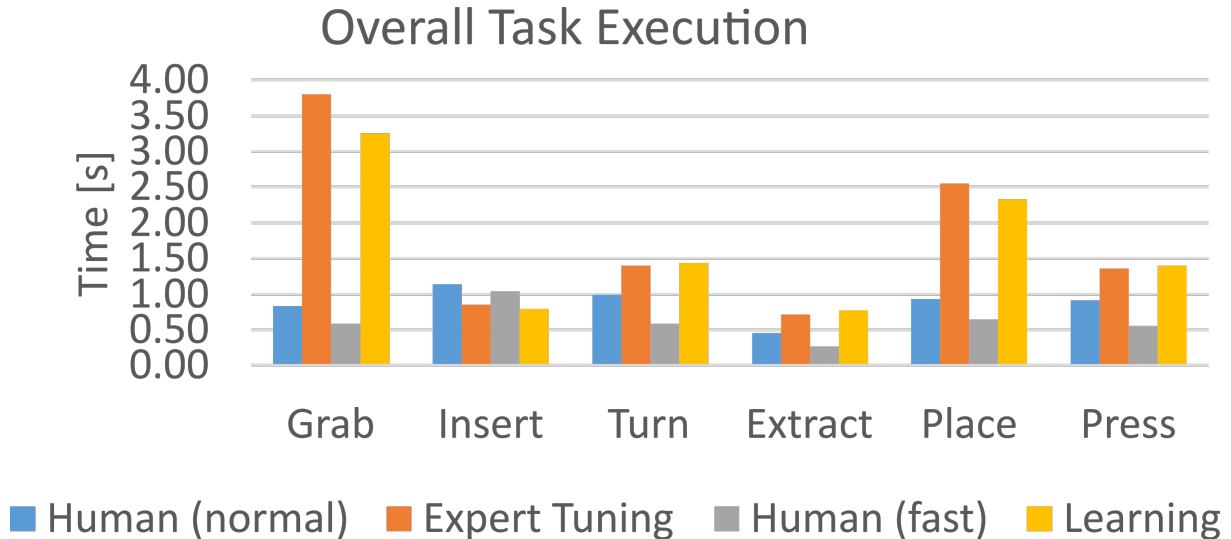
**Comparison** Table 4.7 lists the average execution time and standard deviation for each skill for the robot using expert-tuned parameters and learned parameters as well as the human performance at subjective normal and high speed. The last row shows the required time for the entire task. The success rate is not shown since human and robot (for the case of learning as well as manual tuning) achieved a 100% success rate.

**Table 4.7:** The average execution time and standard deviation in seconds of all skills (executed isolated) and the overall task for human performance (subjective normal and fast speed) and robot performance (expert-tuned and learned). The execution times for skills where human-level performance has been reached are written bold.

Skill	Human (Normal)	Human (Fast)	Robot (Expert)	Robot (Learned)
Grab	$1.2 \pm 0.18$ s	$0.62 \pm 0.13$ s	$3.21 \pm 0.05$ s	$3.08 \pm 0.06$ s
Insertion	$1.43 \pm 0.43$ s	$0.82 \pm 0.2$ s	$0.85 \pm 0.02$ s	<b><math>0.7 \pm 0.07</math> s</b>
Turn	$0.59 \pm 0.19$ s	$0.27 \pm 0.04$ s	$0.61 \pm 0.006$ s	$0.61 \pm 0.005$ s
Extraction	$0.73 \pm 0.24$ s	$0.47 \pm 0.16$ s	<b><math>0.37 \pm 0.004</math> s</b>	<b><math>0.39 \pm 0.003</math> s</b>
Place	$1.53 \pm 0.18$ s	$0.84 \pm 0.2$ s	$2.25 \pm 0.05$ s	$2.04 \pm 0.06$ s
Press Button	$1.02 \pm 0.3$ s	$0.58 \pm 0.22$ s	<b><math>0.53 \pm 0.01</math> s</b>	<b><math>0.6 \pm 0.05</math> s</b>
Complete Task	$5.5 \pm 0.88$ s	$3.73 \pm 0.63$ s	$10.43 \pm 0.3$ s	$10.32 \pm 0.39$ s

Figure 4.16 shows the execution times for each skill when executed not isolated but during the task. To answer the main question behind this case study, i.e. what is the state robotic manipulation performance compared to human manipulation performance:

- It could be shown that state-of-the-art manipulation learning systems on suitable robot hardware are capable of acquiring a range of complex manipulation skills to



**Figure 4.16:** Comparison of skill execution times during task execution.

the same degree of performance as a human expert robot programmer could achieve given the underlying framework is human-understandable.

- Human expert programmers are still faster than learning approaches in programming skills, given a suitable formalism. One reason for this is the human ability of reusing acquired experience. However, note that the employment of robotic solutions in industry and other domains is increasing much more rapidly than the number of available expert programmers.
- The side-by-side comparison clearly demonstrates that current robot manipulation performance still lags behind what humans are capable of.
- The performance gap is mainly due to two factors, i.e. efficient blending of skills and coordinated tactile control, e.g. when grasping or placing an object.
- Considering isolated skill execution, however, robot manipulation performance is catching up and partially even beginning to surpass human performance for difficult skills such as insertion.

**(I)** From this it can be concluded, that future research should also focus on sequencing and blending skills. Although there has been some work on this topic already, e.g. [363], it is mostly concerned with free-space motions and not contact-rich manipulation skills.

**(II)** The results clearly hint at the importance of research into anthropomorphic hand-arm systems, related coordinated tactile control and the potential tactile dexterity and therefore performance increase that may be gained from it, see also [364].

**(III)** Comparing expert tuning and autonomous learning, it is obvious that human experts need significantly less time to tune the skills when they are able to reuse already acquired experience from previous programming. Therefore, transfer learning for real-world manipulation skills is identified as a third important research direction.



## 4.4 Collaborative Assembly Planning

This section presents an experiment to demonstrate the capabilities of the collaborative assembly planner introduced in Sec. 2.3.

### 4.4.1 Experimental Setup

Two kinds of experiments were conducted. First, it was investigated how the planner works on team-level in a realistic setting, however, isolated from real-world effects. In the second experiment the whole framework was tested in a real human-robot collaborative assembly scenario.

To show the output of the team level planner a computational experiment was conducted in which a small assembly, consisting of eight parts is to be assembled by a team of two robots and one human with two different cost metrics. The agents are denoted by  $W = \{r_1, r_2, h\}$ . The corresponding AND/OR graph is shown in Fig. 4.17. None of the atomic parts are in the human worker's reach and the time needed for hand-over actions is estimated to be 8 seconds. Except hand-over actions no interactions are involved.

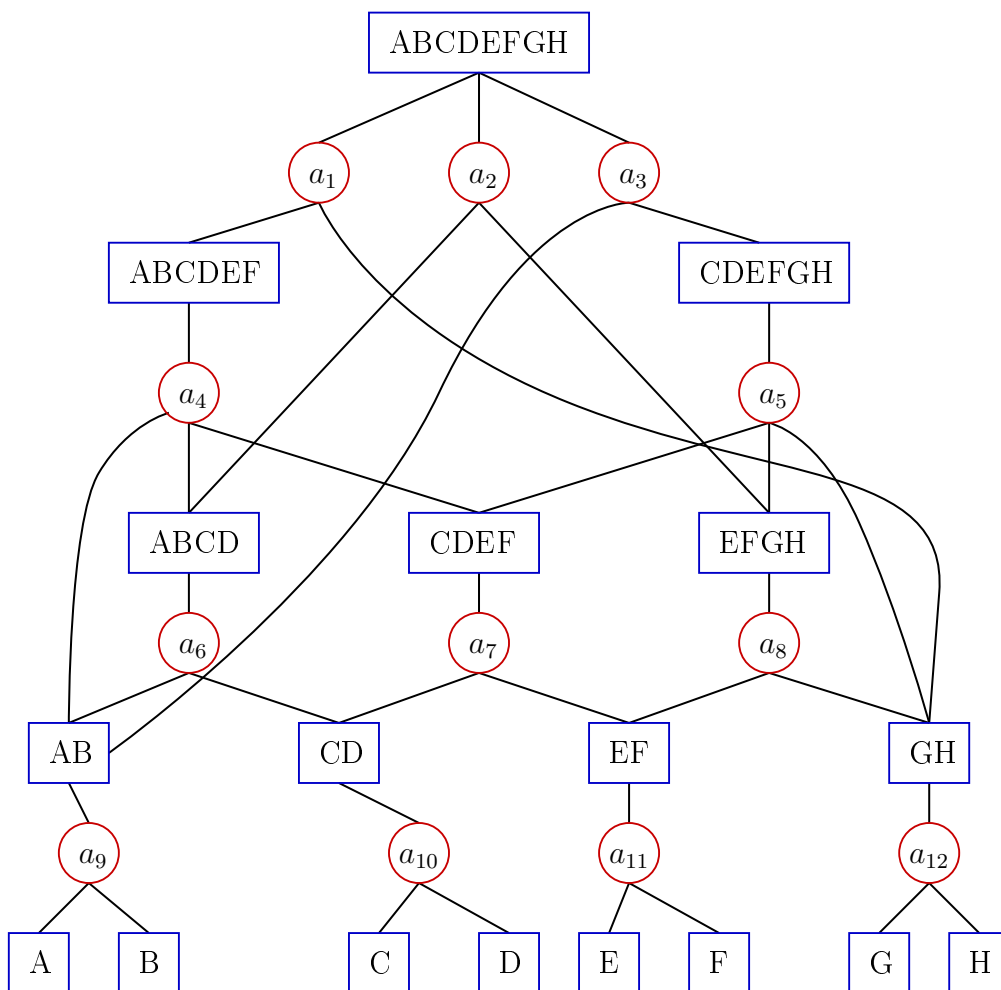
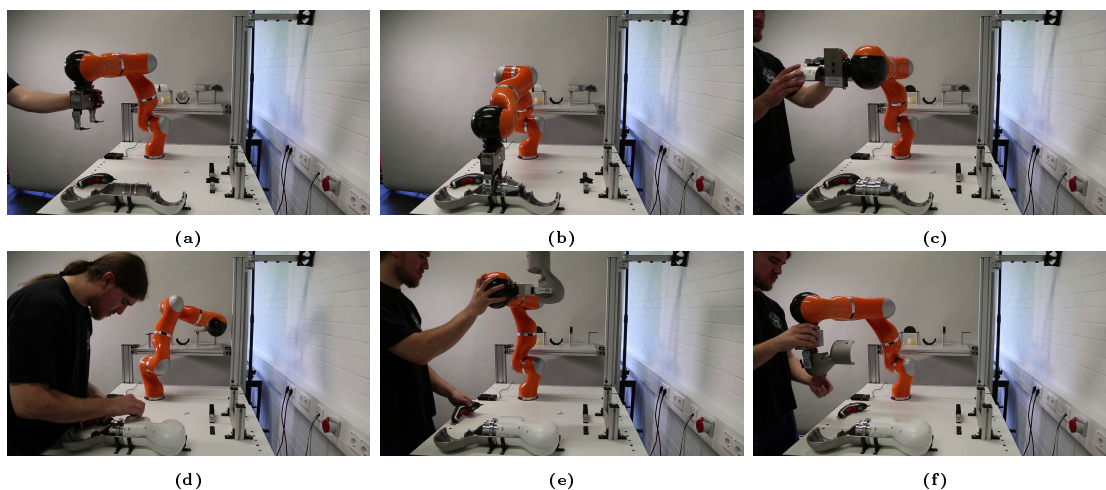


Figure 4.17: AND/OR graph of the experiments assembly plan

As cost function  $c_m = \max_i c(\langle w, a \rangle_i)$  was used.  $\langle w, a \rangle$  denotes the assignment of an agent  $w$  to an action  $a$  in the state  $s$ . The specific cost  $c$  for an assignment  $\langle w, a \rangle$  is listed in Tab 4.8. The left table shows the amount of time the agents need for a specific action, i.e. the cost metric  $C_1$ . In the second cost metric  $C_2$ , human workload is considered as well. Metrics for measuring the human workload can for example be found in [365]. The resulting action sequences are depicted in Fig. 4.19.

The simulation experiments were performed on a computer with a Windows 7 operating system, an Intel i7-3770 processor with 3.4 GHz and 4 GB RAM. The  $A^*$  algorithm took about 40 ms, respectively 2 ms to calculate the action sequences and expanded 74, respectively 6 nodes. A Fibonacci heap was used as data structure for the open list and a hash table for the closed set. In order to indicate the scalability of the presented approach an additional experiment with a strongly connected assembly [114] with 10 parts, two workers and random costs for the worker-actions pairs was conducted. The amount of planning time averages at 15 s. As another example the same experiment was conducted on a binary assembly with 32 parts. All sub assemblies of a binary assembly can be divided into two sub assemblies with an equal number of parts. Here, the average execution time is 3.5 s.

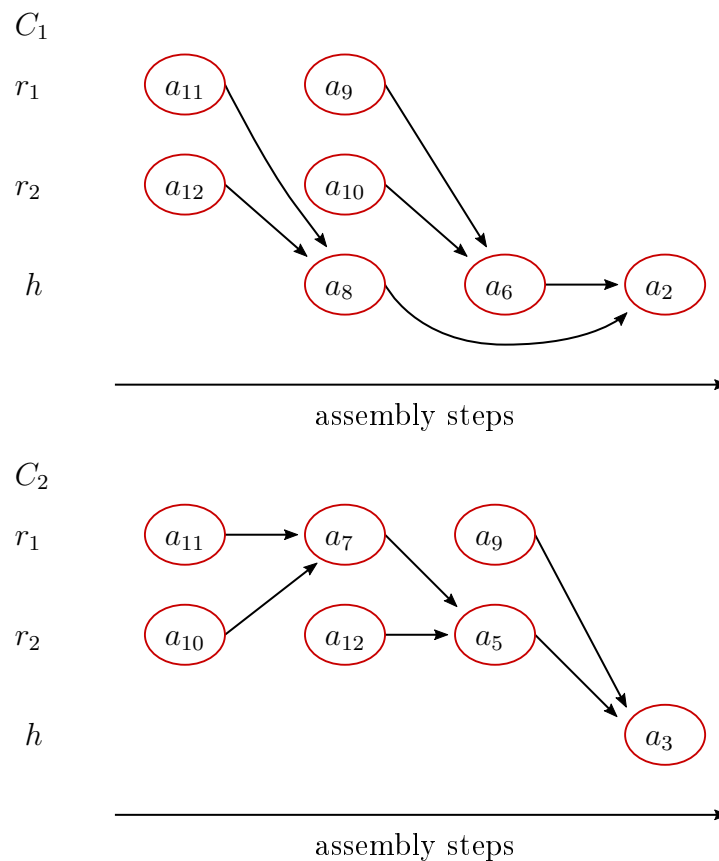
The second experiment (its setting is not related to the previous simulations) was conducted with a real robot in a collaborative assembly scenario, see Fig. 4.18. Note, that for sake of simplicity the actions of the robot and the human in this experiment were not timed (apart from the implicit, discrete timing determined by the team-level). Nonetheless, it is straightforward to introduce an additional timing e.g. via simple human confirmation. Such an input would be incorporated on team-level. Other synchronization schemes that could be employed on agent- or real-time-level are for example time scaling, i.e. the robot would drive slower or even stop in the vicinity of the human. For such capabilities visual perception and/or real-time robot-to-robot communication would have to be available.



**Figure 4.18:** The robot is started by the human (a), the robot uses its assembly skills (b), a tool is handed over to the human (c), the human co-worker and the robot work in parallel (d), the human stops the robot in order to finish an assembly step the robot would otherwise disturb (e), the human takes over an assembly step from the robot manually (f).

**Table 4.8:** Cost Metrics  $C_1$  (top) and  $C_2$  (bottom)

$C_1$	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$	$a_8$	$a_9$	$a_{10}$	$a_{11}$	$a_{12}$
$r_1$	$\infty$	$\infty$	$\infty$	10	5	20	10	$\infty$	20	10	10	10
$r_2$	$\infty$	$\infty$	$\infty$	10	5	10	5	5	20	10	10	10
$h$	20	5	15	20	5	3	15	10	5	5	10	10
$C_2$	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$	$a_8$	$a_9$	$a_{10}$	$a_{11}$	$a_{12}$
$r_1$	$\infty$	$\infty$	$\infty$	10	5	20	10	$\infty$	20	10	10	10
$r_2$	$\infty$	$\infty$	$\infty$	10	5	10	5	5	20	10	10	10
$h$	50	50	50	200	50	30	100	100	50	50	100	100



**Figure 4.19:** Agents' assembly sequences for  $C_1(a)$ , and  $C_2(b)$ . The solid arrows depict a precedence relation, i.e. the source of the arrows provides a needed sub assembly to the sink.

### 4.4.2 Results

It is shown in Fig. 4.19 (top) that the planner produces parallelized execution schemes where possible, which leads to a short overall execution time. A disadvantage of such a parallelized assembly process is the dependency of the agents on each other. If one agent is disturbed in its task, other agents may have to wait. In Fig. 4.19 (bottom) the human

workload has been considered as well, resulting in an execution scheme where the robots work in parallel and the human is only assigned to a task the robots are not capable of performing.

Note, that in order to formally incorporate the aspect of a (human) worker being distracted and therefore potentially disturbing the entire assembly process, one could use a cost function that encodes e.g. some form of worker profile. The probability of a worker being distracted from his task could depend, among others, on his daily routine and experience. The simulation experiments show that the planner can be adapted to the requirements of a specific scenario by providing a cost function that reflects the needs of the situation. E.g. consider that the assembly from Fig. 4.17 is being built in large quantities by human-robot teams. When the demand is normal, a cost metric similar to  $C_2$  could be used. The robots would do most of the work while the human co-workers could be available to other tasks as well. If the demand rises, a cost metric such as  $C_1$  could be used, resulting in a higher production output at higher human workload.

## 4.5 Conclusion

This chapter describes the extensive experimental work done in this thesis. Specifically, the taxonomy of manipulation skills is verified with a large number of challenging skills that exhibit high robustness and performance. This shows that the approach is applicable to a wide range of relevant processes. The learning architecture is experimentally validated with a number of different learning algorithms and a comparison with state-of-the-art deep learning methods. The results aid in selecting compatible learning algorithms and show superior results when compared to the state-of-the-art. Furthermore, a large experimental campaign is described that investigates the architecture's transfer learning capabilities. It demonstrates accelerated learning when reusing knowledge to learn new skills and provides insights into the transfer mechanism such as dependency on geometry and asymmetric transferability. The learning performance as well as achieved manipulation performance were directly compared to human manipulation capabilities and skill programming. The results show that for some skills human performance can already be achieved, while also pointing out the specific gaps that still need to be closed. Finally, a collaborative assembly problem is solved by an automatic planning system and a human-robot team demonstrating how existing skills can be used automatically to solve even complex tasks.

# 5

## Conclusion

### 5.1 Contributions

Future workplaces will make use of tactile robots that need to be equipped with a large variety of manipulation skills in order to cope with the highly dynamic production processes of tomorrow's industry. For example, autonomous factories will produce at lot size one within the context of the production-as-a-service paradigm, meaning that production lines have to be reconfigured regularly. This implies that new processes will constantly introduce new requirements that have to be met by autonomous robot capabilities since human expert programmers will not be able to meet the demand for new skills. The robots will be required to generate optimized skill solutions on-the-fly from simple process descriptions alone. Furthermore, these solutions have to be robust and ideally perform at or above human level. This thesis was inspired by the challenge of providing a way to completely automate the synthesis and optimization of reliable and performant tactile skills for relevant industrial tasks.

An end-to-end manipulation framework has been developed that connects manufacturing processes and tactile skill models with a synthesis pipeline. A learning architecture can directly learn the parameters of those models and optimize them with respect to a given cost function. Thus, the framework is capable of automatically providing optimized tactile skills for given processes without the need for manual programming. This pipeline was validated with extensive experimental work which substantiates the theoretical foundations. Moreover, the experiments yielded sufficient robustness and performance for a number of manipulation skills that are relevant in real-world settings. A comparison study even showcased human-level performance for some cases. Additionally, an assembly planning system was developed to demonstrate the compatibility with advanced AI planning systems. Finally, during the work contained in this thesis, a number of demonstration systems were developed together with a software framework that showcased the various results of this thesis.

**Tactile Skill** A theoretical basis for tactile skills has been laid out that combines torque-controlled hardware systems with suitable tactile controllers, tactile policies and performance evaluation. It defines how tactile stimuli can be perceived and reacted to without necessarily making many assumptions on the used sensors, controllers, and policies. The tactile policies are defined to output energy-encoded commands, i.e. combined twist and wrench. The commands are turned into a desired joint torque by the tactile controller. In general, the tactile skill could make use of various sensors that are capable of perceiving strains and stresses, forces, moments, or external joint torques. However, this thesis focuses on the latter.

**Taxonomy and Synthesis** A taxonomy has been developed to formally organize manipulation processes in a hierarchical structure. Its purpose is to map formal process descriptions to tactile skill models through a synthesis procedure. The procedure selects the models based on process properties such as interaction forces and process steps in an iterative fashion. Each rank of the taxonomy hierarchy applies a selection step to reduce the set of tactile policies that can potentially solve the input process. The output is a single tactile policy that is combined with a suitable controller within a graph-guided twist-wrench policy approximation (GGTWreP) model. The GGTWreP framework was introduced as a way of modeling tactile skills while at the same time complying with process requirements, system limits, and safety. It consists of four layers, namely the semantic layer, the policy layer, the control layer, and the system layer. The semantic layer provides an interface to users and automatic planning systems. It also determines the current state of the skill model by considering the boundary conditions and manipulation steps coming from the process description. The policy layer implements a tactile policy and outputs a twist-wrench command that is fed to the subsequent control layer. The control layer was realized with a unified force / impedance controller. Finally, the system handles the connection to the robot platform, i.e. sends torque commands and receives the percept vector. A large validation experiment confirms the feasibility of the synthesis procedure as well as the GGTWreP framework. The implementation of 28 tactile skills using GGTWreP models exhibits a robust behavior and high performance in various industrial automation tasks subjected to significant process disturbances. Importantly, the simple transfer of policies and the efficient learning through the parameter vector demonstrate the versatility enabled by the taxonomy which enables process experts without specific robotics knowledge to deploy robots in the field using little configuration time.

**Tactile Skill Learning** The output of the synthesis procedure is a tactile skill implemented by a GGTWreP model that can directly be formulated as a learning problem. The skill model has defined parameters with process- and system-informed limits that form a domain on which a learning architecture can autonomously search for an optimal set of parameters that minimize a given cost function such as execution time and contact moments. The learning approach has been experimentally tested on various setups and skills with a strong focus on the insertion skill. The insertion problem (often called peg-in-hole) is representative for very contact-intensive manufacturing processes and therefore highly relevant for real-world industrial processes. A number of algorithms have been compared and it was found that evolutionary strategies such as the covariance matrix adaptation (CMA) method are very suitable for the black-box optimization problem formulated by GGTWreP. Later, an SVM-based algorithm was developed that

improved the learning performance even more. It finds an optimal solution with high robustness by iteratively partitioning the parameter space. The learning architecture was compared to state-of-the-art deep reinforcement learning approaches and outperformed them on the insertion problem both in learning time and final manipulation performance and robustness. During an experiment a transfer effect was observed which led to the hypothesis that GGTWreP is capable of systematically transferring knowledge from one skill to another. The effect was described as robot motor memory (RMM) and connected to the taxonomy hierarchy in terms of skill classes, subclasses, and instances. The effect was verified in a large experimental campaign where nine different insertion skills have been learned and transferred. A total amount of 117.000 single real-world interactions of the robot with the environment have taken place over the course of a week during the completely autonomous experiment. The results clearly show a significant decrease in learning time for similar skills, but also possible adverse effects for specific combinations. An initially suspected linear dependency of the speedup on geometry (e.g. diameter) could not be found. However, a difference at least in terms of geometry classes (e.g. cylinders and keys) was present. Overall, transfer learning enables GGTWreP to scale efficiently to large numbers of skills.

**Tactile Skill Planning** A planning system was devised that is able to allocate a team of robots and humans to tasks in an optimal way such that a given assembly problem is solved. The planner consists of three layers, i.e. the assembly level, the team level, and the agent level. The assembly level models the assembly problem using a modified AND/OR graph. The team level solves the problem of allocating agents (humans or robots) to the actions necessary to build the assembly. The action level connects to tactile skill models that are ordered in a sequence and achieve the planned assembly steps. The system was demonstrated in an experiment with an exemplary assembly where the robot had to pick and place parts, fit them together, and hand them over. The collaborative assembly planner is suited to solve real-world assembly problems by combining the capabilities of humans and robots in an optimal way. The layered architecture enables the system not only to generate nominal plans, but also react to and cope with unforeseen and possibly faulty events, a crucial capability to be prepared for the real-world. The layered planning scheme differentiates between the human-robot team as a whole, the single agent, i.e. every robot and human, and the real-time command structure of that agent. This makes the system able to deal with problems at the right level of abstraction in order to find the most efficient solution, respectively.

## 5.2 Impact

The work in this thesis has had an impact on technological, scientific, and industrial levels. Numerous demonstrators have been developed and showcased at international conferences, trade fairs, and other public events. Some of the most prominent examples are the automatica and Hannover Messe trade fairs, where the learning architecture and tactile skill capabilities have been shown to the highest political level of Germany which led to follow-up visits by e.g. the German chancellor and further investments into the research efforts of the Munich Institute of Robotics and Machine Intelligence. Other significant public events were the official opening of the Munich Institute of Robotics and

Machine Intelligence and the Falling Walls conference. Additionally, parts of this work are used in the labs of Vodafone as part of a telepresence showcase. Further demonstrations at various robotics conferences such as ICRA and IROS disseminated the results into the research community inspiring new work in the fields of manipulation learning and planning. There has been a technology transfer of published algorithms into the industry as well as close collaboration. Also, new projects have been initiated such as the KI.Fabrik, a lighthouse initiative in close cooperation with the industry. Based on the theoretical foundations the Machine Intelligence Operating System(MIOS was developed that enabled many other works at MIRMI and beyond. It powers and enables experimental work on lab automation [366], automatic production [367], learning, control, telepresence [332,346] and human-robot interaction.

### 5.3 Future Work

As a next step, the taxonomy's set of skills should be extended and experimentally validated. The current taxonomy has a strong focus on processes from machine tending and assembly. An extension could focus on material manipulation such as sawing, filing, and grinding. A thorough experimental investigation on those processes would bring new insights that might lead to improvements to the taxonomy hierarchy, the synthesis process, and the GGTWreP model. Moreover, the validation experiments can be improved by considering more realistic process disturbances to test against robustness and performance stability. Besides adding more processes to the taxonomy, it would also be a promising next step to introduce new hardware platforms such as different grippers and arms. This may not only lead to a generalization of the theoretical foundations and capabilities of the overall framework but also allows to include processes in the taxonomy that could so far not be handled due to hardware restrictions. Beyond the extension to more content, the taxonomy may also be adapted to be used as a programming interface. To achieve this, the process definitions have to be formulated as program blocks which can be parameterized by process experts without the need for robot knowledge. Another interesting next step is the use of more complex policies within the GGTWreP models such as dynamic motion primitives (DMP), or neural networks. This could combine very general policy representations with a framework that guarantees stable control and safety. The upscaling of the (transfer) learning capabilities to many robots and skills may accelerate the overall learning performance beyond what is possible today. There have been parallel investigations into collective learning capabilities on the basis of the results of this work. Simulations and experiments indicate significant potential to increase the learning performance for large-scale systems when large numbers of robots share their acquired knowledge. The next steps into this direction involve setting up a large real-world experiment, the necessary software components and devising an experiment procedure. Results from such a system that support the already established theory would have considerable impact on research and in the long-term on commercial robotics products. Finally, the results of this thesis in terms of tactile skill synthesis and learning may be of significant relevance in ongoing research and development in geriatrics. The envisioned humanoid assistants have to rely on a vast range of manipulation skills in order to successfully interact with their highly dynamic and potentially unknown environment. This opens up the possibilities to extend the work of this thesis to other robot platforms (mobile and



bimanual with anthropomorphic hands), and a new domain of processes.





# Appendix

## A.1 Skill Synthesis: Policies

In the following all policies used in the experiments are provided here. Note that for some piece-wise defined policies the conditions for the cases are transitions that activate the respective case, which do not have to remain true once they have been triggered.

$$\begin{aligned}
\pi_{d,1} &= \begin{cases} \dot{\mathbf{x}}_d = f_{lpg}(T_{o_1}, \dot{\mathbf{x}}_g, \ddot{\mathbf{x}}_g) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t, & \pi_{d,2} &= \begin{cases} \dot{\mathbf{x}}_d = f_{rpg}(T_{o_1}, \dot{\mathbf{x}}_g, \ddot{\mathbf{x}}_g) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\pi_{d,3} &= \begin{cases} \dot{\mathbf{x}}_d = f_{pg}(T_{o_1}, \dot{\mathbf{x}}_g, \ddot{\mathbf{x}}_g) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t, & \pi_{d,4} &= \begin{cases} \dot{\mathbf{x}}_d = f_{lpg}(T_{o_1}, \dot{\mathbf{x}}_g, \ddot{\mathbf{x}}_g) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\pi_{d,5} &= \begin{cases} \dot{\mathbf{x}}_d = f_{rpg}(T_{o_1}, \dot{\mathbf{x}}_g, \ddot{\mathbf{x}}_g) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t, & \pi_{d,6} &= \begin{cases} \dot{\mathbf{x}}_d = f_{pg}(T_{o_1}, \dot{\mathbf{x}}_g, \ddot{\mathbf{x}}_g) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\pi_{d,7} &= \begin{cases} \dot{\mathbf{x}}_d = \mathbf{0} \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t, & \pi_{d,8} &= \begin{cases} \dot{\mathbf{x}}_d = \mathbf{0} \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\pi_{d,9} &= \begin{cases} \dot{\mathbf{x}}_d = \mathbf{0} \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\pi}_{d,10} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{lpg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) + \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t) & 0 \end{bmatrix}^T \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,11} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{rpg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) + \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t) & 0 \end{bmatrix}^T \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,12} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) + \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t) & 0 \end{bmatrix}^T \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,13} &= \begin{cases} \dot{\boldsymbol{x}}_d = \mathbf{0} \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,14} &= \begin{cases} \dot{\boldsymbol{x}}_d = \mathbf{0} \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) + \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,15} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{lpg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,16} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{rpg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,17} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,18} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{lpg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) + \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,19} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{rpg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) + \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t \\
\boldsymbol{\pi}_{d,20} &= \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) + \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \quad \forall t
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\pi}_{d,21} &= \begin{cases} \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \emptyset \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \end{cases} \\
\boldsymbol{\pi}_{d,22} &= \begin{cases} \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \emptyset \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \end{cases} \\
\boldsymbol{\pi}_{d,23} &= \begin{cases} \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \emptyset \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_3}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \end{cases} \\
\boldsymbol{\pi}_{d,24} &= \begin{cases} \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \emptyset \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_3}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \end{cases}
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\pi}_{d,25} &= \left\{ \begin{array}{ll} \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } \emptyset \\ \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_3}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \\ \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_4}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } T \in \mathcal{U}(T_{o_3}) \end{array} \right. \\
\boldsymbol{\pi}_{d,26} &= \left\{ \begin{array}{ll} \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } \emptyset \\ \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_3}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \\ \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_4}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } T \in \mathcal{U}(T_{o_3}) \end{array} \right. \\
\boldsymbol{\pi}_{d,27} &= \left\{ \begin{array}{ll} \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } \emptyset \\ \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{array}{l} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_3}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f}t + \boldsymbol{\varphi}) + \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{array} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \end{array} \right.
\end{aligned}$$

$$\begin{aligned}
\pi_{d,28} &= \begin{cases} \dot{\mathbf{x}}_d = f_{pg}(T_{o_1}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } \emptyset \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \\ \dot{\mathbf{x}}_d = f_{pg}(T_{o_2}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \\ \dot{\mathbf{x}}_d = f_{pg}(T_{o_3}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) + \begin{bmatrix} \mathbf{a} \sin(2\pi \mathbf{f} t) & 0 \end{bmatrix}^T & \text{for } \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{contact}} \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{a} \sin(2\pi \mathbf{f} t + \varphi) + \mathbf{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \\
\pi_{d,29} &= \begin{cases} \dot{\mathbf{x}}_d = f_{pg}(T_{o_1}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } \emptyset \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \\ \dot{\mathbf{x}}_d = f_{pg}(T_{o_2}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \\ \dot{\mathbf{x}}_d = f_{pg}(T_{o_3}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) + \begin{bmatrix} \mathbf{a} \sin(2\pi \mathbf{f} t) & 0 \end{bmatrix}^T & \text{for } \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{contact}} \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \\
\pi_{d,30} &= \begin{cases} \dot{\mathbf{x}}_d = f_{pg}(T_{o_1}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } \emptyset \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \\ \dot{\mathbf{x}}_d = f_{pg}(T_{o_2}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \\ \dot{\mathbf{x}}_d = f_{pg}(T_{o_3}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) + \begin{bmatrix} \mathbf{a} \sin(2\pi \mathbf{f} t) & 0 \end{bmatrix}^T & \text{for } \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{contact}} \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} \\
\pi_{d,31} &= \begin{cases} \dot{\mathbf{x}}_d = f_{pg}(T_{o_1}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } \emptyset \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \\ \dot{\mathbf{x}}_d = f_{pg}(T_{o_2}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{contact}} \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \\ \dot{\mathbf{x}}_d = f_{pg}(T_{o_3}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) & \text{for } T \in \mathcal{U}(T_{o_3}) \\ \mathbf{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases}
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\pi}_{d,32} &= \begin{cases} \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \emptyset \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_3}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f} t + \boldsymbol{\varphi}) + \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_4}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } T \in \mathcal{U}(T_{o_3}) \end{cases} \\
\boldsymbol{\pi}_{d,33} &= \begin{cases} \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \emptyset \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_3}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \\ \begin{cases} \dot{\boldsymbol{x}}_d = \mathbf{0} \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \boldsymbol{f}_{\text{ext}} \geq \boldsymbol{f}_{\text{Const.}} \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_4}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } t > t_{\text{wait}} \end{cases} \\
\boldsymbol{\pi}_{d,34} &= \begin{cases} \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) + \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f} t) & 0 \end{bmatrix}^T \\ \boldsymbol{f}_d = \begin{bmatrix} \boldsymbol{a} \sin(2\pi \boldsymbol{f} t + \boldsymbol{\varphi}) + \boldsymbol{f}_{\text{Const.}} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \forall t \\
\boldsymbol{\pi}_{d,35} &= \begin{cases} \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_1}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \mathbf{0} \end{cases} & \text{for } \emptyset \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_2}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \mathbf{0} \end{cases} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{cases} \dot{\boldsymbol{x}}_d = f_{pg}(T_{o_3}, \dot{\boldsymbol{x}}_{g,,} \ddot{\boldsymbol{x}}_{g,,}) \\ \boldsymbol{f}_d = \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{cases} & \text{for } \boldsymbol{f}_{\text{ext}} > \boldsymbol{f}_{\text{contact}} \end{cases}
\end{aligned}$$



$$\pi_{d,36} = \begin{cases} \begin{aligned} \dot{\mathbf{x}}_d &= f_{pg}(T_{o_1}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) \\ \mathbf{f}_d &= \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{aligned} & \text{for } \emptyset \\ \begin{aligned} \dot{\mathbf{x}}_d &= f_{pg}(T_{o_2}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) \\ \mathbf{f}_d &= \begin{bmatrix} \mathbf{0} & f_{\text{grasp}} \end{bmatrix}^T \end{aligned} & \text{for } T \in \mathcal{U}(T_{o_1}) \\ \begin{aligned} \dot{\mathbf{x}}_d &= f_{pg}(T_{o_3}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) \\ \mathbf{f}_d &= \mathbf{0} \end{aligned} & \text{for } \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{contact}} \\ \begin{aligned} \dot{\mathbf{x}}_d &= f_{pg}(T_{o_4}, \dot{\mathbf{x}}_{g,}, \ddot{\mathbf{x}}_{g,}) \\ \mathbf{f}_d &= \mathbf{0} \end{aligned} & \text{for } T \in \mathcal{U}(T_{o_3}) \end{cases}$$



# List of Figures

1.1	The process descriptions that originate e.g. from human technicians are used to synthesize tactile skills based on a taxonomy. The skills are formulated as a learning problem and then optimized by an autonomous learning architecture. Finally, the optimized skills are used in an assembly planning system that solves problems provided by the technician. . . . .	4
1.2	A technological overview of the most relevant currently available robot manipulators, their control paradigms, sensors, interfaces, and target use cases. The robot images are taken from [5, 42–47]. N/A: not applicable; UR: Universal Robot . . . . .	6
1.3	Categorization of transfer learning based on the source and target systems, policy representation, and source and target tasks. $\pi^*$ denotes an optimal policy, $\mathcal{Q}$ a suitable quality metric, and $\theta$ a set of parameters that define the policy $\pi$ . The source system, policy, task space changes, and related work are all categorized. . . . .	17
2.1	Tactile Skill Architecture. $\dot{\mathbf{x}}_d$ is the desired twist, $\mathbf{f}_d$ is the desired wrench, $\boldsymbol{\tau}_d$ is the desired joint torques, $\dot{\mathbf{x}}$ is the actual twist of the robot, $\mathbf{f}_{\text{ext}}$ is the measured external wrench, $\boldsymbol{\Omega}$ is the percept vector, $\mathcal{Q}$ is the performance of the skill, $\theta_\pi$ is the policy parameters, and $\theta_c$ is the controller parameters.	26
2.2	Ranks of the taxonomy of manipulation skills . . . . .	30
2.3	Collaborative assembly planning framework. The top layer depicts the input to the team-level planner, which is an AND/OR graph $Y_{\mathcal{A}}$ resulting from an assembly $\mathcal{A}$ (Step I). The planner then solves the problem of optimal task allocation (Step II) for multiple agents considering a given cost function and constructs a set of assembly sequences (Step III). The actions from the sequences are then passed to the respective agents (Step IV), which possess corresponding skills $\varsigma$ . Skills in turn consist of more basic structures, i.e. atomic actions, which eventually map to the real-time-level.	35
2.4	Partial AND/OR graph of an exemplary assembly. The blue colored rectangles depict OR nodes, the red colored circles AND nodes. . . . .	38
2.5	Example propagation via search through search space. In the top graph agent $w_1$ has been assigned to the indicated AND node in state $\zeta$ and thus, the corresponding assembly action. This AND node has two children that are expanded and form the new state $\zeta'$ . In the bottom graph two parallel actions are assigned, which again yield two children, respectively. The dashed boxes depict the currently active state, the dotted ones the chosen allocation. . . . .	39

2.6	To model an interaction between two agents in the AND/OR graph a new AND/OR node pair is inserted. In this example, the subassembly $\Gamma_k$ is created by agent $w_1$ via joining $\Gamma_i$ and $\Gamma_j$ , which corresponds to the action $a_1 := a(\text{assemble}, w_a)$ . Yet, some kind of interaction is necessary to complete the assembly step. E.g., the performing agent cannot reach $P_i$ so another agent $w_2$ needs to hand this part over. Therefore, $\Gamma'_i$ and the AND node that corresponds to the interaction $a_2 := a(\text{hand\_over}, (w_1, w_2))$ are inserted on the right side. . . . .	41
2.7	Monotonically decreasing cost function, learning effort LE, and learning effort ratio LER . . . . .	49
2.8	Transfer learning observation with a 10x reduction in the learning time for task B. $\theta_0$ denotes the initial parameters drawn from a normal distribution $\mathcal{N}(\cdot)$ with a mean $\mu$ and standard deviation $\sigma$ . $\theta_A^*$ and $\theta_B^*$ are the optimal parameters for tasks A and B, respectively. . . . .	50
2.9	Hierarchical three-level classification proposal of tactile manipulation skills	51
3.1	Architectural overview of the GGTWreP framework . . . . .	54
3.2	The two examples show the process description and their corresponding GGTWreP models. . . . .	57
3.3	Overview of the MIOS modules . . . . .	60
3.4	Overview and working principle of the learning module . . . . .	66
3.5	(Top left) The field of dentronics [331]. (Bottom left) The modalities of the devised interaction framework dependent on the communication range [15]. (Right) Experimental prototype setup for the conducted user-study consisting of a Franka Emika Robot arm [5] (a) equipped with an SR300 camera [341] (b), a mobile device (c), colored gloves (d), a pedal (e) and a microphone (f) [15]. . . . .	67
3.6	(Top left) The teleoperation setup consists of a leader (LR), a follower (FR) and an human operator robot (HO) which is rigidly connected to the leader for providing repeatable experimental scenarios [332]. (Top right) The kickoff of the KI.Fabrik project © TUM. (Bottom) The one-to-many telepresence showcase in the collective © TUM. . . . .	69
3.7	KI.ROBOTIC.DESIGN in the Pinakothek der Moderne © TUM . . . . .	70
3.8	(Left) The collaborative assembly station [14]. (Right) The later version of the station at the visit of chancellor Angela Merkel © TUM. . . . .	71
3.9	From left to right and top to bottom: A political event at the monastery in Seon © CSU, the opening of the Vodafone 5G lab © Vodafone, Hannover Messe 2019 © TUM, DLD 2019 © TUM, AI Council of Bavaria © TUM. . . . .	71
3.10	From left to right and top to bottom: The official opening of the Munich Institute of Robotics and Machine Intelligence (MIRMI) © TUM, the visit of German chancellor Angela Merkel at MIRMI © TUM, the AI.BAY conference 2023 © TUM, Falling Walls 2019 © TUM. . . . .	72
4.1	Taxonomy of manipulation skills, experimentally validated skills are shown with the used setup. For clarity, the taxonomy ranks are indicated as family (F), domain (D), class (C), and subclass (S). Instances are omitted since they are represented by the pictures. . . . .	77

4.2	A comparison of the required energy to learn a great number of skills. The deep deterministic policy gradient (DDPG) algorithm is compared with the GGTWreP framework (see Sec. 3.2.5 both with and without transfer learning. . . . .	80
4.3	Experimental setup, puzzle (top right), key (bottom left) and cylinder (bottom right) . . . . .	81
4.4	Experimental results. The columns correspond to the learning algorithms (LHS, BO, PSO, CMA-ES) and the rows to the tasks (Puzzle, Key, Cylinder). The blue line denotes the cost of an algorithm/task combination averaged over 10 experiments. The light blue area denotes the 90 % confidence interval. . . . .	83
4.5	Experiment setup consisting of a robot arm and a key/lock combination . .	84
4.6	Results of the comparison experiment for HiREPS, CMA-ES, and PSP. The blue line denotes the cost averaged over 10 experiments. The light blue area denotes the 90 % confidence interval. . . . .	85
4.7	Results of the comparison experiment for A3C, DDPG, SAC, and T3D . .	85
4.8	The analyzed reference tasks consist of six cylinders with increasing diameters and three different common household keys (bottom left). Five robots were used in total for the experiment. . . . .	87
4.9	<b>Top)</b> The transfer learning performance of DDPG. <b>Bottom)</b> A representation of the SAC algorithm’s learning performance in the insertion task. Graphs are shown for the 30 mm, 40 mm, and 60 mm pegs. The rewards above the green line indicate successful trials. For the 30 mm and the 40 mm pegs, no successful trials were observed. While successful trials were observed for the 60 mm peg, these were random and not systemically exploited. . . . .	91
4.10	The logarithmic cost evolution of learning a single skill without prior knowledge (solid blue lines) and the mean cost when using a level-0 (dashed lines), level-1 (dotted lines), and level-2 transfer (dash-dotted lines), respectively. Each task was learned using prior knowledge from every other task. For clarity, the results from each subclass (cylinders and keys) were grouped by averaging the resulting costs and plotting the mean. The cylinder insertion tasks are $\iota_1$ to $\iota_6$ , and the key insertion tasks are $\iota_7$ to $\iota_9$ . . . . .	92
4.11	Average learning success rate (ALSR) in the same pattern as Figure 4.10. For better readability, the ALSR was approximated by time-window averaging and plotted as stair function. The black dashed line is the theoretical optimal case with a slope of 1. For each skill, the average ALSR is shown when taking prior knowledge from the cylinder tasks and from the key. Note that an $ALSR < 1$ means that not enough learning time has passed to fully converge to a feasible solution. However, the difference between raw learning and transfer learning over time is of interest. . . . .	93
4.12	<b>Top)</b> The ET, speedup factor, and LER for each task combination. The black dividers separate the level-0, level-1, and level-2 transfers (see the legend on the right). On the left side, the target tasks are denoted, and, at the bottom, the source tasks. The color maps are depicted to the right of each plot. <b>Bottom)</b> The mean LER, ET, and speedup factor ( $\log_{10}$ ) over all tasks for the respective transfer levels. . . . .	94

4.13	The intention of the case study is to draw a comparison between robot and human manipulation performance. This also entails a comparison between the learning and tuning approaches towards this performance level. For the presented case study a number of skills of varying tactile complexity was selected and a reference task was constructed based on them. . . . .	95
4.14	The left picture shows the overall setup. The right side shows the respective steps of the task. The arrows indicate the direction of movement for the task steps: 1. Grab the key, 2. Insert the key, 3. Turn the key and turn in back, 4. Extract the key, 5. Place the key, 6. Press the button. . . . .	96
4.15	Learning process for all skills. The blue line denotes the cost over time averaged over 10 experiments. The light blue area denotes the 95% confidence interval. The red triangles denote the tuning steps of the expert programmer. . . . .	99
4.16	Comparison of skill execution times during task execution. . . . .	100
4.17	AND/OR graph of the experiments assembly plan . . . . .	101
4.18	The robot is started by the human (a), the robot uses its assembly skills (b), a tool is handed over to the human (c), the human co-worker and the robot work in parallel (d), the human stops the robot in order to finish an assembly step the robot would otherwise disturb (e), the human takes over an assembly step from the robot manually (f). . . . .	102
4.19	Agents' assembly sequences for $C_1(a)$ , and $C_2(b)$ . The solid arrows depict a precedence relation, i.e. the source of the arrows provides a needed sub assembly to the sink. . . . .	103

# List of Tables

2.1	Suitability of existing learning algorithms with respect to the desired properties . . . . .	44
4.1	Setup descriptions . . . . .	78
4.2	Experimental results for all skills . . . . .	79
4.3	Optimized costs for successful trials . . . . .	85
4.4	Successful trials over episodes in % (averaged over experiments) . . . . .	86
4.5	DDPG parameters . . . . .	88
4.6	Parameter domains . . . . .	97
4.7	The average execution time and standard deviation in seconds of all skills (executed isolated) and the overall task for human performance (subjective normal and fast speed) and robot performance (expert-tuned and learned). The execution times for skills where human-level performance has been reached are written bold. . . . .	99
4.8	Cost Metrics $C_1$ (top) and $C_2$ (bottom) . . . . .	103





# Bibliography

- [1] G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl, “Dlr’s torque-controlled light weight robot iii—are we reaching the technological limits now?” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2002, pp. 1710–1716.
- [2] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, “The dlr lightweight robot: design and control concepts for robots in human environments,” *Industrial Robot: An International Journal*, vol. 34, no. 5, pp. 376–385, 2007.
- [3] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, *et al.*, “The kuka-dlr lightweight robot arm—a new reference platform for robotics research and manufacturing,” in *Proc. International Symposium on Robotics (ISR) and German Conference on Robotics (ROBOTIK)*. VDE, 2010, pp. 1–8.
- [4] A. Campeau-Lecours, H. Lamontagne, S. Latour, P. Fauteux, V. Maheu, F. Boucher, C. Deguire, and L.-J. C. L’Ecuyer, “Kinova modular robot arms for service robotics applications,” in *Rapid Automation: Concepts, Methodologies, Tools, and Applications*. IGI global, 2019, pp. 693–719.
- [5] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, M. Sabaghian, C. Jaehne, L. Hausperger, and S. Haddadin, “The franka emika robot: A reference platform for robotics research and education,” *Robotics & Automation Magazine (RAM)*, pp. 2–20, 2022.
- [6] N. Hogan, “Impedance control: An approach to manipulation: Part i—theory,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–7, 1985.
- [7] A. Albu-Schaffer and G. Hirzinger, “Cartesian impedance control techniques for torque controlled light-weight robots,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2002, pp. 657–663.
- [8] A. Albu-Schäffer, C. Ott, and G. Hirzinger, “A unified passivity-based control framework for position, torque and impedance control of flexible joint robots,” *The International Journal of Robotics Research (IJRR)*, vol. 26, no. 1, pp. 23–39, 2007.
- [9] Y. Li, G. Ganesh, N. Jarrassé, S. Haddadin, A. Albu-Schaeffer, and E. Burdet, “Force, impedance, and trajectory learning for contact tooling and haptic identification,” *Transactions on Robotics (T-RO)*, vol. 34, no. 5, pp. 1170–1182, 2018.

- [10] A. de Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, O. Wilde, Ed. [Place of publication not identified]: John Wiley, 2007, pp. 1623–1630.
- [11] S. Haddadin, A. de Luca, and A. Albu-Schaffer, "Robot collisions: A survey on detection, isolation, and identification," *Transactions on Robotics (T-RO)*, vol. 33, no. 6, pp. 1292–1312, 2017.
- [12] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.
- [13] M. Indri, A. Grau, and M. Ruderman, "Guest editorial special section on recent trends and developments in industry 4.0 motivated robotic solutions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1677–1680, 2018.
- [14] L. Johannsmeier and S. Haddadin, "A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes," *Robotics and Automation Letters (R-AL)*, vol. 2, no. 1, pp. 41–48, 2017.
- [15] J. Grischke, L. Johannsmeier, L. Eich, and S. Haddadin, "Dentronics: Review, first concepts and pilot study of a new application domain for collaborative robots in dental assistance," in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6525–6532.
- [16] V. Seidita, F. Lanza, A. Pipitone, and A. Chella, "Robots as intelligent assistants to face covid-19 pandemic," *Briefings in Bioinformatics*, vol. 22, no. 2, pp. 823–831, 2021.
- [17] E. Martinez-Martin and A. P. del Pobil, "Personal robot assistants for elderly care: an overview," *Personal Assistants: Emerging Computational Technologies*, pp. 77–91, 2018.
- [18] M. Tröbinger, C. Jähne, Z. Qu, J. Elsner, A. Reindl, S. Getz, T. Goll, B. Loinger, T. Loibl, C. Kugler, *et al.*, "Introducing garmin-a service robotics platform to support the elderly at home: Design philosophy, system overview and first results," *Robotics and Automation Letters (R-AL)*, vol. 6, no. 3, pp. 5857–5864, 2021.
- [19] M. V. Zinggeler, "The educational duty of the german chamber of commerce," *Global Business Languages*, vol. 7, no. 1, p. 9, 2010.
- [20] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 156–163.
- [21] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

- [22] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, *The Computational Limits of Deep Learning*, 2020.
- [23] L. Johannsmeier, M. Gerchow, and S. Haddadin, “A framework for robot manipulation: Skill formalism, meta learning and adaptive control,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5844–5850.
- [24] L. Johannsmeier, S. Schneider, F. Voigt, and S. Haddadin, “Motor memory for few-shot learning in robotic manipulation,” *submission in preparation*, 2024.
- [25] L. Johannsmeier, S. Schneider, Y. Li, E. Burdet, and S. Haddadin, “A process-centric manipulation taxonomy for the organisation, classification and synthesis of tactile robot skills,” *submission in preparation*, 2024.
- [26] A. de la Garza, “A revolutionary robotic arm: The 50 best inventions of 2018,” 2018. [Online]. Available: <https://time.com/collection-post/5454734/panda/>
- [27] S. Haddadin, A. Albu-Schaffer, M. Frommberger, J. Rossmann, and G. Hirzinger, “The “dlr crash report”: Towards a standard crash-testing protocol for robot safety - part i: Results,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 272–279.
- [28] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, “Requirements for safe robots: Measurements, analysis and new insights,” *The International Journal of Robotics Research (IJRR)*, vol. 28, no. 11-12, pp. 1507–1527, 2009.
- [29] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, “Rlbench: The robot learning benchmark & learning environment,” *Robotics and Automation Letters (R-AL)*, vol. 5, no. 2, pp. 3019–3026, 2020.
- [30] N. Jaquier, L. Roza, S. Calinon, and M. Bürger, “Bayesian optimization meets riemannian manifolds in robot learning,” in *Proc. Conference on Robot Learning (CoRL)*, 2020, pp. 233–246.
- [31] A. S. Chen, H. Nam, S. Nair, and C. Finn, “Batch exploration with examples for scalable robotic reinforcement learning,” *Robotics and Automation Letters (R-AL)*, vol. 6, no. 3, pp. 4401–4408, 2021.
- [32] L. Roveda, M. Magni, M. Cantoni, D. Piga, and G. Bucca, “Assembly task learning and optimization through human’s demonstration and machine learning,” in *Proc. International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 1852–1859.
- [33] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, “Closing the sim-to-real loop: Adapting simulation randomization with real world experience,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8973–8979.
- [34] B. Rooks, “The harmonious robot,” *Industrial Robot: An International Journal*, vol. 33, no. 2, pp. 125–130, 2006.

- [35] C. Fitzgerald, “Developing baxter,” in *Proc. Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2013, pp. 1–6.
- [36] Rethink Robotics, “Sawyer collaborative robots for industrial automation,” 2015. [Online]. Available: <https://www.rethinkrobotics.com/sawyer>
- [37] KUKA AG, “Mensch-roboter-kollaboration - der lbr iiwa - kuka ag,” 06.02.2021. [Online]. Available: <https://www.kuka.com/de-de/produkte-leistungen/robotersysteme/industrieroboter/lbr-iiwa>
- [38] R. Treffler, “"ich bin der neue",” *KUKA AG*, 31.01.2019. [Online]. Available: <https://www.blog.kuka.com/2019/01/31/lbr-iisy/>
- [39] F.-P. Kirgis, P. Katsos, and M. Kohlmaier, “Collaborative robotics,” in *Robotic Fabrication in Architecture, Art and Design*, D. Reinhardt, R. Saunders, and J. Burry, Eds. Springer International Publishing, 2016, pp. 448–453.
- [40] Universal Robots, “Collaborative robotic automation | cobots from universal robots,” 06.02.2021. [Online]. Available: <https://www.universal-robots.com/>
- [41] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer, “Towards autonomous robotic butlers: Lessons learned with the pr2,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 5568–5575.
- [42] Universal Robots. (2008) Ur series. [Online]. Available: <https://robots.ieee.org/robots/universal/>
- [43] ABB. (2011) yumi. [Online]. Available: <https://robots.ieee.org/robots/yumi/>
- [44] Barret Technnology. (2004) Wam. [Online]. Available: <https://robots.ieee.org/robots/wam/>
- [45] KUKA. (2013) Kuka iiwa. [Online]. Available: <https://robots.ieee.org/robots/lbriiwa/>
- [46] Rethink Robotics. (2015) Sawyer. [Online]. Available: <https://robots.ieee.org/robots/sawyer/>
- [47] W. Kreft, “Inverse kinematics determination and trajectory tracking algorithm development of a robot with 7 joints,” in *Proc. International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2020, pp. 1001–1007.
- [48] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schaeffer, and E. Burdet, “Human-like adaptation of force and impedance in stable and unstable interactions,” *Transactions on Robotics (T-RO)*, vol. 27, no. 5, pp. 918–930, 2011.
- [49] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger, “Cartesian impedance control of redundant robots: recent results with the dlr-light-weight-arms,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2003, pp. 3704–3709.

- [50] S. A. B. Birjandi, J. Kühn, and S. Haddadin, “Observer-extended direct method for collision monitoring in robot manipulators using proprioception and imu sensing,” *Robotics and Automation Letters (R-AL)*, vol. 5, no. 2, pp. 954–961, 2020.
- [51] N. Hogan, “Impedance control: An approach to manipulation,” in *1984 American Control Conference*. IEEE, 6/6/1984 - 6/8/1984, pp. 304–313.
- [52] —, “Impedance control: An approach to manipulation: Part ii—implementation,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 8–16, 1985.
- [53] R. J. Anderson and M. W. Spong, “Hybrid impedance control of robotic manipulators,” *Journal on Robotics and Automation*, vol. 4, no. 5, pp. 549–556, 1988.
- [54] N. Hogan, “Stable execution of contact tasks using impedance control,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1987, pp. 1047–1054.
- [55] D. A. Lawrence, “Impedance control stability properties in common implementations,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1988, pp. 1185–1190.
- [56] S. Schneider and R. H. Cannon, “Object impedance control for cooperative manipulation: theory and experimental results,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1989, pp. 1076–1083.
- [57] S. Jung, T. C. Hsia, and R. G. Bonitz, “Force tracking impedance control of robot manipulators under unknown environment,” *Transactions on Control Systems Technology*, vol. 12, no. 3, pp. 474–483, 2004.
- [58] G. Ferretti, G. Magnani, and P. Rocco, “Impedance control for elastic joints industrial manipulators,” *Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 488–498, 2004.
- [59] V. Duchaine and C. M. Gosselin, “General model of human-robot cooperation using a novel velocity based variable impedance control,” in *Proc. Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. IEEE, 2007, pp. 446–451.
- [60] C. Ott, *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*, ser. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2008.
- [61] C. Ott, A. Albu-Schaffer, A. Kugi, and G. Hirzinger, “On the passivity-based impedance control of flexible joint robots,” *Transactions on Robotics (T-RO)*, vol. 24, no. 2, pp. 416–429, 2008.
- [62] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, “Learning variable impedance control,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

- [63] P. Song, Y. Yu, and X. Zhang, “Impedance control of robots: An overview,” in *International Conference on Cybernetics, Robotics and Control (CRC)*. IEEE, 2017, pp. 51–55.
- [64] ———, “A tutorial survey and comparison of impedance control on robotic manipulation,” *Robotica*, vol. 37, no. 5, pp. 801–836, 2019.
- [65] J. C. P. Ibarra and A. A. Siqueira, “Impedance control of rehabilitation robots for lower limbs, review,” in *Proc. Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol*. IEEE, 2014, pp. 235–240.
- [66] Q. P. Ha, Q. H. Nguyen, D. C. Rye, and H. F. Durrant-Whyte, “Impedance control of a hydraulically actuated robotic excavator,” *Automation in Construction*, vol. 9, no. 5-6, pp. 421–435, 2000.
- [67] J. Koivumaki and J. Mattila, “Stability-guaranteed impedance control of hydraulic robotic manipulators,” *Transactions on Mechatronics*, vol. 22, no. 2, pp. 601–612, 2017.
- [68] A. Toedtheide, T. Lilge, and S. Haddadin, “Antagonistic impedance control for pneumatically actuated robot joints,” *Robotics and Automation Letters (R-AL)*, vol. 1, no. 1, pp. 161–168, 2016.
- [69] A. Toedtheide, E. Shahriari, and S. Haddadin, “Tank based unified torque/impedance control for a pneumatically actuated antagonistic robot joint,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1255–1262.
- [70] A. Dietrich, T. Wimbock, and A. Albu-Schaffer, “Dynamic whole-body mobile manipulation with a torque controlled humanoid robot via impedance control laws,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011, pp. 3199–3206.
- [71] A. Dietrich, K. Bussmann, F. Petit, P. Kotyczka, C. Ott, B. Lohmann, and A. Albu-Schäffer, “Whole-body impedance control of wheeled mobile manipulators,” *Autonomous Robots*, vol. 40, no. 3, pp. 505–517, 2016.
- [72] K. Bussmann, A. Dietrich, and C. Ott, “Whole-body impedance control for a planetary rover with robotic arm: Theory, control design, and experimental validation,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 910–917.
- [73] A. Rocchi, E. M. Hoffman, D. G. Caldwell, and N. G. Tsagarakis, “Opensot: A whole-body control library for the compliant humanoid robot coman,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6248–6253.
- [74] J. Vorndamme, M. Schappler, A. Todtheide, and S. Haddadin, “Soft robotics for the hydraulic atlas arms: Joint impedance control with collision detection and disturbance compensation,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3360–3367.

- [75] V. Lippiello and F. Ruggiero, "Exploiting redundancy in cartesian impedance control of uavs equipped with a robotic arm," in *Proc. International Conference on Intelligent Robots and Systems (IROS)*, I. Staff, Ed. IEEE, 2012, pp. 3768–3773.
- [76] A. M. Khan, D.-w. Yun, M. A. Ali, J. Han, K. Shin, and C. Han, "Adaptive impedance control for upper limb assist exoskeleton," in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4359–4366.
- [77] Z. Li, Z. Huang, W. He, and C.-Y. Su, "Adaptive impedance control for an upper limb robotic exoskeleton using biological signals," *Transactions on Industrial Electronics*, vol. 64, no. 2, pp. 1664–1674, 2017.
- [78] L. J. Love and W. J. Book, "Force reflecting teleoperation with adaptive impedance control," *Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 1, pp. 159–165, 2004.
- [79] E. Nuno, R. Ortega, N. Barabanov, and L. Basanez, "A globally stable pd controller for bilateral teleoperators," *Transactions on Robotics (T-RO)*, vol. 24, no. 3, pp. 753–758, 2008.
- [80] M. Tufail and C. W. de Silva, "Impedance control schemes for bilateral teleoperation," in *Proc. International Conference on Computer Science & Education*. IEEE, 2014, pp. 44–49.
- [81] T. Wimbock, B. Jahn, and G. Hirzinger, "Synergy level impedance control for multifingered hands," in *International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 973–979.
- [82] R. Carelli and R. Kelly, "An adaptive impedance/force controller for robot manipulators," *Transactions on Automatic Control*, vol. 36, no. 8, pp. 967–971, 1991.
- [83] R. Kelly, R. Carelli, M. Amestegui, and R. Ortega, "On adaptive impedance control of robot manipulators," in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1989, pp. 572–577.
- [84] R. Ikeura and H. Inooka, "Variable impedance control of a robot for cooperation with a human," in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1995, pp. 3097–3102.
- [85] R. Colbaugh, H. Seraji, and K. Glass, "Direct adaptive impedance control of robot manipulators," *Journal of Robotic Systems*, vol. 10, no. 2, pp. 217–248, 1993.
- [86] R. Kamnik, D. Matko, and T. Bajd, "Application of model reference adaptive control to industrial robot impedance control," *Journal of Intelligent and Robotic Systems*, vol. 22, no. 2, pp. 153–163, 1998.
- [87] C.-C. Cheah and D. Wang, "Learning impedance control for robotic manipulators," *Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 452–465, 1998.
- [88] M. Sharifi, S. Behzadipour, and G. Vossoughi, "Nonlinear model reference adaptive impedance control for human–robot interactions," *Control Engineering Practice*, vol. 32, pp. 9–27, 2014.

- [89] E. Burdet, R. Osu, D. W. Franklin, T. E. Milner, and M. Kawato, “The central nervous system stabilizes unstable dynamics by learning optimal impedance,” *Nature*, vol. 414, no. 6862, pp. 446–449, 2001.
- [90] G. Ganesh, A. Albu-Schaffer, M. Haruno, M. Kawato, and E. Burdet, “Biomimetic motor behavior for simultaneous adaptation of force, impedance and trajectory in interaction tasks,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 2705–2711.
- [91] G. Ganesh, N. Jarrassé, S. Haddadin, A. Albu-Schäffer, and E. Burdet, “A versatile biomimetic controller for contact tooling and haptic exploration,” in *Proc. International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3329–3334.
- [92] O. Khatib and J. Burdick, “Motion and force control of robot manipulators,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1986, pp. 1381–1386.
- [93] D. E. Whitney, “Historical perspective and state of the art in robot force control,” *The International Journal of Robotics Research*, vol. 6, no. 1, pp. 3–14, 1987.
- [94] S. Eppinger and W. Seering, “On dynamic models of robot force control,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1986, pp. 29–34.
- [95] ———, “Understanding bandwidth limitations in robot force control,” in *Proc. International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, 1987, pp. 904–909.
- [96] K. S. Eom, I. H. Suh, W. K. Chung, and S.-R. Oh, “Disturbance observer based force control of robot manipulator without force sensor,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1998, pp. 3012–3017.
- [97] J. Roy and L. L. Whitcomb, “Adaptive force control of position/velocity controlled robots: theory and experiment,” *Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 121–137, 2002.
- [98] R. Cortesao and M. Dominici, “Robot force control on a beating heart,” *Transactions on Mechatronics*, vol. 22, no. 4, pp. 1736–1743, 2017.
- [99] B. J. Stephens and C. G. Atkeson, “Dynamic balance force control for compliant humanoid robots,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 1248–1255.
- [100] E. Magrini, F. Flacco, and A. de Luca, “Control of generalized contact motion and force in physical human-robot interaction,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2298–2304.
- [101] Z. Li, B. Wang, F. Sun, C. Yang, Q. Xie, and W. Zhang, “semg-based joint force control for an upper-limb power-assist exoskeleton robot,” *Journal of Biomedical and Health Informatics*, vol. 18, no. 3, pp. 1043–1050, 2014.



- [102] S. Haddadin, S. Haddadin, and S. Parusel, “Franka emika gmbh,” 2017. [Online]. Available: [www.franka.de](http://www.franka.de)
- [103] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson, “Force controlled robotic assembly without a force sensor,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 1538–1543.
- [104] G. Zeng and A. Hemami, “An overview of robot force control,” *Robotica*, vol. 15, no. 5, pp. 473–482, 1997.
- [105] B. Siciliano and L. Villani, *Robot Force Control*, ser. Kluwer international series in engineering and computer science: Robotics: vision, manipulation, and sensors. Springer US, 1999.
- [106] T. Yoshikawa, “Force control of robot manipulators,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2000, pp. 220–226.
- [107] F. Almeida, A. Lopes, and P. Abreu, “Force-impedance control: a new control strategy of robotic manipulators,” *Recent Advances in Mechatronics*, vol. 1, pp. 126–137, 1999.
- [108] C. Schindlbeck and S. Haddadin, “Unified passivity-based cartesian force/impedance control for rigid and flexible joint robots via task-energy tanks,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 440–447.
- [109] K. Karacan, H. Sadeghian, R. Kirschner, and S. Haddadin, “Passivity-based skill motion learning in stiffness-adaptive unified force-impedance control,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 9604–9611.
- [110] A. G. Marin and R. Weitschat, “Unified impedance and hybrid force-position controller with kinestatic filtering,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3353–3359.
- [111] Y. Lin, Z. Chen, and B. Yao, “Unified motion/force/impedance control for manipulators in unknown contact environments based on robust model-reaching approach,” *Transactions on Mechatronics*, vol. 26, no. 4, pp. 1905–1913, 2021.
- [112] P. Zech, E. Renaudo, S. Haller, X. Zhang, and J. Piater, “Action representations in robotics: A taxonomy and systematic classification,” *The International Journal of Robotics Research (IJRR)*, vol. 38, no. 5, pp. 518–562, 2019.
- [113] A. Chella, M. Frixione, and S. Gaglio, “A conceptual representation of the actions of an autonomous robot,” in *Proc. European Workshop on Advanced Mobile Robots (Eurobot)*. IEEE, 1999, pp. 97–104.
- [114] L. S. Homem de Mello and A. C. Sanderson, “And/or graph representation of assembly plans,” *Transactions on Robotics and Automation*, vol. 6, no. 2, pp. 188–199, 1990.

- [115] T. Cao and A. C. Sanderson, “And/or net representation for robotic task sequence planning,” *Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 28, no. 2, pp. 204–218, 1998.
- [116] J. Yang, Y. Xu, and C. S. Chen, “Human action learning via hidden markov model,” *Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 27, no. 1, pp. 34–44, 1997.
- [117] M. N. Nicolescu and M. J. Matarić, “A hierarchical architecture for behavior-based robots,” in *Proc. International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. ACM Press, 2002, p. 227.
- [118] Y. Demiris and M. Johnson, “Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning,” *Connection Science*, vol. 15, no. 4, pp. 231–243, 2003.
- [119] W. Erhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. van Schie, and H. Bekkering, “Goal-directed imitation for robots: A bio-inspired approach to action understanding and skill learning,” *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 353–360, 2006.
- [120] R. Zoliner, M. Pardowitz, S. Knoop, and R. Dillmann, “Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2005, pp. 1535–1540.
- [121] V. Kruger, D. L. Herzog, S. Baby, A. Ude, and D. Kragic, “Learning actions from observations,” *Robotics & Automation Magazine (RAM)*, vol. 17, no. 2, pp. 30–43, 2010.
- [122] B. J. Cohen, S. Chitta, and M. Likhachev, “Search-based planning for manipulation with motion primitives,” in *Proc. International Conference on Robotics and Automation (ICRA)*. Piscataway, N.J.: IEEE, 2010, pp. 2902–2908.
- [123] T. Cao and A. C. Sanderson, “Task decomposition and analysis of robotic assembly task plans using petri nets,” *Transactions on Industrial Electronics*, vol. 41, no. 6, pp. 620–630, 1994.
- [124] P. Lima, H. Gracio, V. Veiga, and A. Karlsson, “Petri nets for modeling and coordination of robotic tasks,” in *SMC’98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*. IEEE, 1998, pp. 190–195.
- [125] H. Costelha and P. Lima, “Modelling, analysis and execution of robotic tasks using petri nets,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2007, pp. 1449–1454.
- [126] —, “Robot task plan representation by petri nets: modelling, identification, analysis and execution,” *Autonomous Robots*, vol. 33, no. 4, pp. 337–360, 2012.

- [127] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ogren, “Towards a unified behavior trees framework for robot control,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5420–5427.
- [128] F. Rovida, B. Grossmann, and V. Kruger, “Extended behavior trees for quick definition of flexible robotic tasks,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6793–6800.
- [129] B. Banerjee, “Autonomous acquisition of behavior trees for robot control,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3460–3467.
- [130] K. French, S. Wu, T. Pan, Z. Zhou, and O. C. Jenkins, “Learning behavior trees from demonstration,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7791–7797.
- [131] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krueger, and F. Wörgötter, “Object action complexes as an interface for planning and robot control,” in *Proc. International Conference on Humanoid Robots (Humanoids)*. IEEE, 2006.
- [132] D. Kraft, N. Pugeault, E. Başeski, M. Popović, D. Kragić, S. Kalkan, F. Wörgötter, and N. Krüger, “Birth of the object: Detection of objectness and extraction of object shape through object–action complexes,” *International Journal of Humanoid Robotics*, vol. 05, no. 02, pp. 247–265, 2008.
- [133] K. Huebner, M. Björkman, B. Rasolzadeh, M. Schmidt, and D. Kragic, “Integration of visual and shape attributes for object action complexes,” in *Proc. International Conference on Computer Vision Systems (ICVS)*. Springer-Verlag Berlin Heidelberg, 2008, pp. 13–22.
- [134] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen, *et al.*, “A formal definition of object-action complexes and examples at different levels of the processing hierarchy,” *PACO-PLUS Technical Repor*, 2009.
- [135] F. Wörgötter, A. Agostini, N. Krüger, N. Shylo, and B. Porr, “Cognitive agents — a procedural perspective relying on the predictability of object-action-complexes (oacs),” *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 420–432, 2009.
- [136] M. Wachter, S. Schulz, T. Asfour, E. Aksoy, F. Worgotter, and R. Dillmann, “Action sequence reproduction based on automatic segmentation and object-action complexes,” in *Proc. International Conference on Humanoid Robots (Humanoids)*. IEEE, 2013, pp. 189–195.
- [137] M. J. Aein, E. E. Aksoy, M. Tamosiunaite, J. Papon, A. Ude, and F. Worgotter, “Toward a library of manipulation actions based on semantic object-action relations,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 4555–4562.

- [138] M. Do, J. Schill, J. Ernesti, and T. Asfour, “Learn to wipe: A case study of structural bootstrapping from sensorimotor experience,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1858–1864.
- [139] M. Tenorth and M. Beetz, “A unified representation for reasoning about robot actions, processes, and their effects on objects,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 1351–1358.
- [140] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, “Representation and exchange of knowledge about actions, objects, and environments in the roboearth framework,” *Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 643–651, 2013.
- [141] A. Wahrburg, S. Zeiss, B. Matthias, J. Peters, and H. Ding, “Combined pose-wrench and state machine representation for modeling robotic assembly skills,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 852–857.
- [142] M. Diab, M. Pomarlan, D. Beßler, A. Akbari, J. Rosell, J. Bateman, and M. Beetz, “Skillman — a skill-based robotic manipulation framework based on perception and reasoning,” *Robotics and Autonomous Systems*, vol. 134, p. 103653, 2020.
- [143] A. Cangelosi, G. Metta, G. Sagerer, S. Nolfi, C. Nehaniv, K. Fischer, J. Tani, T. Belpaeme, G. Sandini, F. Nori, L. Fadiga, B. Wrede, K. Rohlfing, E. Tuci, K. Dautenhahn, J. Saunders, and A. Zeschel, “Integration of action and language knowledge: A roadmap for developmental robotics,” *Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 167–195, 2010.
- [144] C. G. Atkeson and S. Schaal, “Learning tasks from a single demonstration,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1997, pp. 1706–1712.
- [145] S. Vijayakumar and S. Schaal, “Locally weighted projection regression: An  $o(n)$  algorithm for incremental real time learning in high dimensional space,” in *Proc. International Conference on Machine Learning (ICML)*, vol. 1, 2000, pp. 288–293.
- [146] S. Schaal, S. Kotosaka, and D. Sternad, “Nonlinear dynamical systems as movement primitives,” in *Proc. International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2000, pp. 1–11.
- [147] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 63–71.
- [148] M. Schneider and W. Ertel, “Robot learning by demonstration with local gaussian process regression,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 255–260.
- [149] D. Nguyen-Tuong, M. Seeger, and J. Peters, “Model learning with local gaussian process regression,” *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

- [150] Y. Engel, P. Szabo, and D. Volkinshtein, “Learning to control an octopus arm with gaussian process temporal difference methods,” *Advances in Neural Information Processing Systems*, vol. 18, pp. 347–354, 2005.
- [151] R. S. Sutton and A. G. Barto, *Reinforcement Learning, Second Edition: An Introduction*. MIT Press, 2018.
- [152] George Konidaris, Sarah Osentoski, and Philip Thomas, “Value function approximation in reinforcement learning using the fourier basis,” *Proc. Conference on Artificial Intelligence (AAAI)*, vol. 25, no. 1, 2011.
- [153] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, “Robot learning from demonstration by constructing skill trees,” *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 3, pp. 360–375, 2012.
- [154] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.
- [155] S. Niu, S. Chen, H. Guo, C. Targonski, M. Smith, and J. Kovačević, “Generalized value iteration networks: Life beyond lattices,” in *Proc. Conference on Artificial Intelligence (AAAI)*, vol. 32, 2018.
- [156] F. Diaz Ledezma and S. Haddadin, “First-order-principles-based constructive network topologies: An application to robot inverse dynamics,” in *Proc. International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017.
- [157] S. Schaal, “Dynamic movement primitives -a framework for motor control in humans and humanoid robotics,” in *Adaptive Motion of Animals and Machines*. Springer Tokyo, 2006, pp. 261–280.
- [158] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: learning attractor models for motor behaviors,” *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.
- [159] T. Matsubara, S.-H. Hyon, and J. Morimoto, “Learning parametric dynamic movement primitives from multiple demonstrations,” *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.
- [160] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, “Incremental learning of full body motion primitives and their sequencing through human motion observation,” *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 3, pp. 330–345, 2012.
- [161] D. Lee and C. Ott, “Incremental kinesthetic teaching of motion primitives using the motion refinement tube,” *Autonomous Robots*, vol. 31, no. 2-3, pp. 115–131, 2011.
- [162] B. Nemeč and A. Ude, “Action sequencing using dynamic movement primitives,” *Robotica*, vol. 30, no. 5, pp. 837–846, 2012.

- [163] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proc. International Conference on Humanoid Robots (Humanoids)*. IEEE, 2008, pp. 91–98.
- [164] A. Namiki and S. Yokosawa, "Robotic origami folding with dynamic motion primitives," in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5623–5628.
- [165] A. Mitrevski, A. Padalkar, M. Nguyen, and P. G. Plöger, "'lucy, take the noodle box!': Domestic object manipulation using movement primitives and whole body motion," in *Proc. CoboCup*. Springer, 2019, pp. 189–200.
- [166] B. Nemeč, A. Gams, M. Denisa, and A. Ude, "Human-robot cooperation through force adaptation using dynamic motion primitives and iterative learning," in *Proc. International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2014, pp. 1439–1444.
- [167] Platt, Robert, Jr., R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Proc. Robotics: Science and Systems (RSS)*, 2010.
- [168] T. Cavalier-Smith, "A revised six-kingdom system of life," *Biological Reviews*, vol. 73, no. 3, pp. 203–266, 1998.
- [169] G. L. Snider, "Nosology for our day: its application to chronic obstructive pulmonary disease," *American Journal of Respiratory and Critical Care Medicine*, vol. 167, no. 5, pp. 678–683, 2003.
- [170] M. J. Adams, E. J. Lefkowitz, A. M. Q. King, and E. B. Carstens, "Recently agreed changes to the international code of virus classification and nomenclature," *Archives of Virology*, vol. 158, no. 12, pp. 2633–2639, 2013.
- [171] M. A. Park, *Introducing anthropology: An integrated approach*, 2nd ed. Boston: McGraw-Hill, 2003.
- [172] Delphi Group *et al.*, "Information intelligence: content classification and the enterprise taxonomy practice," 2004.
- [173] D. A. Wiegmann and S. A. Shappell, *A human error approach to aviation accident analysis: The human factors analysis and classification system*. Routledge, 2017.
- [174] A. Bloomfield, Y. Deng, J. Wampler, P. Rondot, D. Harth, M. McManus, and N. Badler, "A taxonomy and comparison of haptic actions for disassembly tasks," in *Proc. Virtual Reality*. IEEE, 2003, pp. 225–231.
- [175] J. O. Huckaby and H. I. Christensen, "A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics," in *Proc. Conference on Artificial Intelligence (AAAI)*, 2012.

- [176] D. Leidner, C. Borst, A. Dietrich, M. Beetz, and A. Albu-Schaffer, “Classifying compliant manipulation tasks for automated planning in robotics,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1769–1776.
- [177] A. Bjorkelund, L. Edstrom, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Storkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx, “On the integration of skilled robot motions for productivity in manufacturing,” in *Proc. International Symposium on Assembly and Manufacturing (ISAM)*. IEEE, 2011, pp. 1–9.
- [178] J. Pfrommer, M. Schleipen, and J. Beyerer, “Pprs: Production skills and their relation to product, process, and resource,” in *Proc. Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2013, pp. 1–4.
- [179] A. Huamán Quispe, H. Ben Amor, and H. I. Christensen, “A taxonomy of benchmark tasks for robot manipulation,” in *Robotics Research: Volume 1*. Springer International Publishing, 2018, pp. 405–421.
- [180] D. Paulius, Y. Huang, J. Meloncon, and Y. Sun, “Manipulation motion taxonomy and coding for robots,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5596–5601.
- [181] M. R. Cutkosky, “On grasp choice, grasp models, and the design of hands for manufacturing tasks,” *Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989.
- [182] I. M. Bullock, R. R. Ma, and A. M. Dollar, “A hand-centric classification of human and robot dexterous manipulation,” *Transactions on Haptics*, vol. 6, no. 2, pp. 129–144, 2013.
- [183] J. Liu, F. Feng, Y. C. Nakamura, and N. S. Pollard, “A taxonomy of everyday grasps in action,” in *Proc. International Conference on Humanoid Robots (Humanoids)*. IEEE, 2014, pp. 573–580.
- [184] J. R. Napier, “The prehensile movements of the human hand,” *The Journal of Bone and Joint Surgery*, vol. 38-B, no. 4, pp. 902–913, 1956.
- [185] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, “A taxonomy for swarm robots,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1993, pp. 441–447.
- [186] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *The International Journal of Robotics Research (IJRR)*, vol. 23, no. 9, pp. 939–954, 2004.
- [187] G. A. Korsah, A. Stentz, and M. B. Dias, “A comprehensive taxonomy for multi-robot task allocation,” *The International Journal of Robotics Research (IJRR)*, vol. 32, no. 12, pp. 1495–1512, 2013.

- [188] J. Borras and T. Asfour, “A whole-body pose taxonomy for loco-manipulation tasks,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1578–1585.
- [189] J. Borràs, C. Mandery, and T. Asfour, “A whole-body support pose taxonomy for multi-contact humanoid robot motions,” *Science Robotics*, vol. 2, no. 13, 2017.
- [190] K. Tsiakas, M. Kyrarini, V. Karkaletsis, F. Makedon, and O. Korn, “A taxonomy in robot-assisted training: Current trends, needs and challenges,” in *Proc. Pervasive Technologies Related to Assistive Environments Conference*, vol. 6, no. 4, 2018, p. 119.
- [191] M. Linjawi and R. K. Moore, “Towards a comprehensive taxonomy for characterizing robots,” in *Proc. Towards Autonomous Robotic Systems*. Springer International Publishing, 2018, pp. 381–392.
- [192] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A survey of socially interactive robots,” *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 143–166, 2003.
- [193] J. Bautista-Ballester, J. Vergés-Llahí, and D. Puig, “Programming by demonstration: A taxonomy of current relevant methods to teach and describe new skills to robots,” in *First Iberian Robotics Conference: Advances in Robotics (ROBOT)*. Springer International Publishing, 2014, vol. 252, pp. 287–300.
- [194] L. Onnasch and E. Roesler, “A taxonomy to structure and analyze human–robot interaction,” *International Journal of Social Robotics*, vol. 13, no. 4, pp. 833–849, 2021.
- [195] R. E. Fikes, P. E. Hart, and N. J. Nilsson, “Learning and executing generalized robot plans,” *Artificial Intelligence*, vol. 3, pp. 251–288, 1972.
- [196] B. Horne, M. Jamshidi, and N. Vadiie, “Neural networks in robotics: A survey,” *Journal of Intelligent and Robotic Systems*, vol. 3, no. 1, pp. 51–66, 1990.
- [197] K. L. Moore, M. Dahleh, and S. P. Bhattacharyya, “Iterative learning control: A survey and new results,” *Journal of Robotic Systems*, vol. 9, no. 5, pp. 563–594, 1992.
- [198] J. H. Connell and S. Mahadevan, “Introduction to robot learning,” *Robot Learning*, pp. 1–17, 1993.
- [199] P. Bakker and Y. Kuniyoshi, “Robot see, robot do: An overview of robot imitation,” in *Proc. Workshop on Learning in Robots and Animals (AISB)*, 1996, pp. 3–11.
- [200] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
- [201] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.



- [202] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1238–1274, 2013.
- [203] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
- [204] A. S. Polydoros and L. Nalpantidis, “Survey of model-based reinforcement learning: Applications on robotics,” *Journal of Intelligent and Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.
- [205] Z. Zhu and H. Hu, “Robot learning from demonstration in robotic assembly: A survey,” *Robotics*, vol. 7, no. 2, p. 17, 2018.
- [206] H. Nguyen and H. La, “Review of deep reinforcement learning for robot manipulation,” in *Proc. International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 590–595.
- [207] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.
- [208] W. Zhao, J. P. Queralta, and T. Westerlund, “Sim-to-real transfer in deep reinforcement learning for robotics: a survey,” in *Proc. Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 737–744.
- [209] O. Kroemer, S. Niekum, and G. Konidaris, *A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms*. Journal of Machine Learning Research, 2021.
- [210] J. Cui and J. Trinkle, “Toward next-generation learned robot manipulation,” *Science Robotics*, vol. 6, no. 54, 2021.
- [211] S. Xiao, Z. Wang, and J. Folkesson, “Unsupervised robot learning to predict person motion,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 691–696.
- [212] C. Finn, I. Goodfellow, and S. Levine, “Unsupervised learning for physical interaction through video prediction,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [213] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta, “The curious robot: Learning visual representations via physical interactions,” in *Proc. European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2016, pp. 3–18.
- [214] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, “Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning,” *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 13-14, pp. 1595–1618, 2017.

- [215] L.-J. Lin, “Hierarchical learning of robot skills by reinforcement,” in *Proc. International Conference on Neural Networks*. IEEE, 1993, pp. 181–186.
- [216] Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei, “Surreal: Open-source reinforcement learning framework and robot manipulation benchmark,” in *Proc. Conference on Robot Learning (CoRL)*, 2018, pp. 767–782.
- [217] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, “Learning modular neural network policies for multi-task and multi-robot transfer,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2169–2176.
- [218] L. Rozo, N. Jaquier, S. Calinon, and D. G. Caldwell, “Learning manipulability ellipsoids for task compatibility in robot manipulation,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3183–3189.
- [219] J. Kober and J. R. Peters, “Policy search for motor primitives in robotics,” in *Proc. Advances in Neural Information Processing Systems*, 2009, pp. 849–856.
- [220] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, “Skill learning and task outcome prediction for manipulation,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3828–3834.
- [221] P. Kormushev, S. Calinon, and D. Caldwell, “Reinforcement learning in robotics: Applications and real-world challenges,” *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.
- [222] O. Kroemer, C. Daniel, G. Neumann, H. van Hoof, and J. Peters, “Towards learning hierarchical skills for multi-phase manipulation tasks,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1503–1510.
- [223] A. K. Tanwani and S. Calinon, “Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model,” *Robotics and Automation Letters (R-AL)*, vol. 1, no. 1, pp. 235–242, 2016.
- [224] C. Devin, P. Abbeel, T. Darrell, and S. Levine, “Deep object-centric representations for generalizable robot learning,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7111–7118.
- [225] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, “Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks,” in *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 12627–12637.
- [226] W. Montgomery, A. Ajay, C. Finn, P. Abbeel, and S. Levine, “Reset-free guided policy search: Efficient deep reinforcement learning with stochastic initial states,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3373–3380.
- [227] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, “Composable deep reinforcement learning for robotic manipulation,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6244–6251.

- [228] P. Ennen, P. Bresenitz, R. Vossen, and F. Hees, “Learning robust manipulation skills with guided policy search via generative motor reflexes,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7851–7857.
- [229] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Bjorkman, “Deep predictive policy training using reinforcement learning,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2351–2358.
- [230] T. Inoue, G. de Magistris, A. Munawar, T. Yokoya, and R. Tachibana, “Deep reinforcement learning for high precision assembly tasks,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 819–825.
- [231] A. Rupam Mahmood, D. Korenkevych, B. J. Komer, and J. Bergstra, “Setting up a reinforcement learning task with a real-world robot,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4635–4640.
- [232] S. J. Pan and Q. Yang, “A survey on transfer learning,” *Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [233] K. Weiss, T. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big Data*, vol. 3, 2016.
- [234] M. E. Taylor and P. Stone, “Transfer learning for reinforcement learning domains: A survey,” *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.
- [235] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *Proc. International Conference on Artificial Neural Networks*, 2018, pp. 270–279.
- [236] F. L. Da Silva and A. H. R. Costa, “A survey on transfer learning for multiagent reinforcement learning systems,” *Journal of Artificial Intelligence Research*, vol. 64, pp. 645–703, 2019.
- [237] S. Chernova and A. L. Thomaz, “Robot learning from human teachers,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.
- [238] K. Kronander and A. Billard, “Learning compliant manipulation through kinesthetic and tactile human-robot interaction,” *Transactions on Haptics*, vol. 7, no. 3, pp. 367–380, 2014.
- [239] F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, “Force-based variable impedance learning for robotic manipulation,” *Robotics and Autonomous Systems*, vol. 109, pp. 156–167, 2018.
- [240] M. Hueser, T. Baier, and J. Zhang, “Learning of demonstrated grasping skills by stereoscopic tracking of human head configuration,” in *Proc. International Conference on Robotics and Automation, 2006. ICRA 2006*. IEEE, 2006, pp. 2795–2800.

- [241] Yezhou Yang, Yi Li, Cornelia Fermuller, and Yiannis Aloimonos, “Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015. [Online]. Available: <https://ojs.aaai.org/index.php/aaai/article/view/9671>
- [242] C. Yang, C. Zeng, C. Fang, W. He, and Z. Li, “A dmps-based framework for robot learning and generalization of humanlike variable impedance skills,” *Transactions on Mechatronics*, vol. 23, no. 3, pp. 1193–1203, 2018.
- [243] M. Wigness, J. G. Rogers, and L. E. Navarro-Serment, “Robot navigation from human demonstration: Learning control behaviors,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1150–1157.
- [244] E. Pignat and S. Calinon, “Learning adaptive dressing assistance from human demonstration,” *Robotics and Autonomous Systems*, vol. 93, pp. 61–75, 2017.
- [245] J. Kim, N. Cauli, P. Vicente, B. Damas, F. Cavallo, and J. Santos-Victor, ““icub, clean the table!” a robot learning from demonstration approach using deep neural networks,” in *Proc. International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2018, pp. 3–9.
- [246] S. Barrett, M. E. Taylor, and P. Stone, “Transfer learning for reinforcement learning on a physical robot,” in *Proc. International Conference on Autonomous Agents and Multiagent Systems-Adaptive Learning Agents Workshop (AAMAS-ALA)*, vol. 1, 2010.
- [247] M. K. Helwa and A. P. Schoellig, “Multi-robot transfer learning: A dynamical system perspective,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4702–4708.
- [248] E. Heiden, L. Palmieri, L. Bruns, K. O. Arras, G. S. Sukhatme, and S. Koenig, “Bench-mr: A motion planning benchmark for wheeled mobile robots,” *Robotics and Automation Letters (R-AL)*, vol. 6, no. 3, pp. 4536–4543, 2021.
- [249] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.
- [250] J. Van Baar, A. Sullivan, R. Cordorel, D. Jha, D. Romeres, and D. Nikovski, “Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6001–6007.
- [251] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, “Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach,” *Applied Sciences*, vol. 10, no. 19, p. 6923, 2020.
- [252] P. D. H. Nguyen, T. Fischer, H. J. Chang, U. Pattacini, G. Metta, and Y. Demiris, “Transferring visuomotor learning from simulation to the real world for robotics

- manipulation tasks,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6667–6674.
- [253] A. A. Rusu, M. Večer\`ik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, “Sim-to-real robot learning from pixels with progressive nets,” in *Proc. Conference on Robot Learning (CoRL)*, 2017, pp. 262–270.
- [254] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer, “Sim-to-real transfer with neural-augmented robot simulation,” in *Proc. Conference on Robot Learning (CoRL)*, 2018, pp. 817–828.
- [255] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.
- [256] J. Matas, S. James, and A. J. Davison, “Sim-to-real reinforcement learning for deformable object manipulation,” in *Proc. Conference on Robot Learning (CoRL)*, 2018, pp. 734–743.
- [257] J. H. Metzen, A. Fabisch, L. Senger, J. Gea Fernández, and E. A. Kirchner, “Towards learning of generic skills for robotic manipulation,” *KI - Künstliche Intelligenz*, vol. 28, no. 1, pp. 15–20, 2014.
- [258] J. van Baar, A. Sullivan, R. Cordorel, D. Jha, D. Romeres, and D. Nikovski, “Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6001–6007.
- [259] M. Lutter, S. Mannor, J. Peters, D. Fox, and A. Garg, “Robust value iteration for continuous control tasks,” *arXiv preprint arXiv:2105.12189*, 2021.
- [260] Ahmed H Qureshi, Arsalan Mousavian, Chris Paxton, Michael Yip, and Dieter Fox, “Nerp: Neural rearrangement planning for unknown objects,” in *Proc. Robotics: Science and Systems (RSS)*, 2021.
- [261] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters, “Learning to play table tennis from scratch using muscular robots,” *Transactions on Robotics (T-RO)*, 2022.
- [262] L. Sievers, J. Pitz, and B. Bäuml, “Learning purely tactile in-hand manipulation with a torque-controlled hand,” *arXiv preprint arXiv:2204.03698*, 2022.
- [263] B. Nemeč, F. J. Abu-Dakka, B. Ridge, A. Ude, J. A. Jorgensen, T. R. Savarimuthu, J. Jouffroy, H. G. Petersen, and N. Kruger, “Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile,” in *Proc. International Conference on Advanced Robotics (ICAR)*, 2013, pp. 1–7.
- [264] R. Laha, L. F. Figueredo, J. Vrabel, A. Swikir, and S. Haddadin, “Reactive cooperative manipulation based on set primitives and circular fields,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6577–6584.

- [265] Nemanja Rakicevic and Petar Kormushev, “Active learning via informed search in movement parameter space for efficient robot task learning and transfer,” *Autonomous Robots*, vol. 43, no. 8, pp. 1917–1935, 2019.
- [266] H. Bruyninckx, “Open robot control software: the orocos project,” in *Proc. International Conference on Robotics and Automation (ICRA)*, vol. 3. IEEE, 2001, pp. 2523–2528.
- [267] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, “Ros: an open-source robot operating system,” in *Proc. International Conference on Robotics and Automation - Workshop on Open Source Software*, vol. 3, no. 3.2. IEEE, 2009, p. 5.
- [268] Y. Maruyama, S. Kato, and T. Azumi, “Exploring the performance of ros2,” in *Proc. Conference on Embedded Software*, 2016, pp. 1–10.
- [269] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. R. Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, *et al.*, “ros\_control: A generic and simple control framework for ros,” *The Journal of Open Source Software*, vol. 2, no. 20, pp. 456–456, 2017.
- [270] M. Li, Z. Cai, X. Yi, Z. Wang, Y. Wang, Y. Zhang, and X. Yang, “Alliance-ros: a software architecture on ros for fault-tolerant cooperative multi-robot systems,” in *Proc. Pacific Rim International Conference on Artificial Intelligence (PRICAI)*. Springer, 2016, pp. 233–242.
- [271] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour, “The robot software framework armarx,” *IT-Information Technology*, vol. 57, no. 2, pp. 99–111, 2015.
- [272] L. Muratore, A. Laurenzi, E. M. Hoffman, A. Rocchi, D. G. Caldwell, and N. G. Tsagarakis, “Xbotcore: A real-time cross-robot software platform,” in *Proc. International Conference on Robotic Computing (IRC)*. IEEE, 2017, pp. 77–80.
- [273] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoğlu, and G. Bartels, “Know rob 2.0 - a 2nd generation knowledge processing framework for cognition-enabled robotic agents,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 512–519.
- [274] A. BAUER, D. WOLLHERR, and M. BUSS, “Human–robot collaboration: A survey,” *International Journal of Humanoid Robotics*, vol. 05, no. 01, pp. 47–66, 2008.
- [275] M. A. Goodrich and A. C. Schultz, *Human-Robot Interaction: A Survey*, ser. Foundations and trends in human-computer interaction. Now Publishers, 2008.
- [276] B. Chandrasekaran and J. M. Conrad, “Human-robot collaboration: A survey,” in *Proc. SoutheastCon*. IEEE, 2015, pp. 1–8.
- [277] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, “Progress and prospects of the human–robot collaboration,” *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.

- [278] B. D. Argall and A. G. Billard, “A survey of tactile human–robot interactions,” *Robotics and Autonomous Systems*, vol. 58, no. 10, pp. 1159–1176, 2010.
- [279] V. Villani, F. Pini, F. Leali, and C. Secchi, “Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications,” *Mechatronics*, vol. 55, pp. 248–266, 2018.
- [280] I. Maurtua, A. Ibarguren, J. Kildal, L. Susperregi, and B. Sierra, “Human–robot collaboration in industrial applications,” *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, p. 172988141771601, 2017.
- [281] J. Arents, V. Abolins, J. Judvaitis, O. Vismanis, A. Oraby, and K. Ozols, “Human–robot collaboration trends and safety aspects: A systematic review,” *Journal of Sensor and Actuator Networks*, vol. 10, no. 3, p. 48, 2021.
- [282] M. Valori, A. Scibilia, I. Fassi, J. Saenz, R. Behrens, S. Herbster, C. Bidard, E. Lucet, A. Magisson, L. Schaake, J. Bessler, G. B. Prange-Lasonder, M. Kühnrich, A. B. Lassen, and K. Nielsen, “Validating safety in human–robot collaboration: Standards and new perspectives,” *Robotics*, vol. 10, no. 2, p. 65, 2021.
- [283] J. Rosell, “Assembly and task planning using petri nets: A survey,” *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 218, no. 8, pp. 987–994, 2004.
- [284] X. Jia and M. Q.-H. Meng, “A survey and analysis of task allocation algorithms in multi-robot systems,” in *Proc. International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2013, pp. 2280–2285.
- [285] S. Alartartsev, S. Stellmacher, and F. Ortmeier, “Robotic task sequencing problem: A survey,” *Journal of Intelligent and Robotic Systems*, vol. 80, no. 2, pp. 279–298, 2015.
- [286] A. Khamis, A. Hussein, and A. Elmogy, “Multi-robot task allocation: A review of the state-of-the-art,” in *Proc. Cooperative Robots and Sensor Networks*, A. Koubâa and J. Martínez-de Dios, Eds. Springer International Publishing, 2015, pp. 31–51.
- [287] P. Tsarouchi, S. Makris, and G. Chryssolouris, “Human–robot interaction review and challenges on task planning and programming,” *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 8, pp. 916–931, 2016.
- [288] L. Wang, R. Gao, J. Vánca, J. Krüger, X. V. Wang, S. Makris, and G. Chryssolouris, “Symbiotic human-robot collaborative assembly,” *CIRP Annals*, vol. 68, no. 2, pp. 701–726, 2019.
- [289] W. Zhang, “Representation of assembly and automatic robot planning by petri net,” *Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 2, pp. 418–422, 1989.
- [290] M. Bonert, L. Shu, and B. Benhabib, “Motion planning for multi-robot assembly systems,” *International Journal of Computer Integrated Manufacturing*, vol. 13, no. 4, pp. 301–310, 2000.

- [291] G. Michalos, S. Makris, J. Spiliotopoulos, I. Misios, P. Tsarouchi, and G. Chrysolouris, “Robo-partner: Seamless human-robot cooperation for intelligent, flexible and safe operations in the assembly factories of the future,” *Procedia CIRP*, vol. 23, pp. 71–76, 2014.
- [292] K. Darvish, F. Wanderlingh, B. Bruno, E. Simetti, F. Mastrogiovanni, and G. Casalino, “Flexible human–robot cooperation models for assisted shop-floor tasks,” *Mechatronics*, vol. 51, pp. 97–114, 2018.
- [293] K.-C. Ying, P. Pourhejazy, C.-Y. Cheng, and C.-H. Wang, “Cyber-physical assembly system-based optimization for robotic assembly sequence planning,” *Journal of Manufacturing Systems*, vol. 58, pp. 452–466, 2021.
- [294] I. Rodríguez, K. Nottensteiner, D. Leidner, M. Durner, F. Stulp, and A. Albu-Schäffer, “Pattern recognition for knowledge transfer in robotic assembly sequence planning,” *Robotics and Automation Letters (R-AL)*, vol. 5, no. 2, pp. 3666–3673, 2020.
- [295] F. McGlone, J. Wessberg, and H. Olausson, “Discriminative and affective touch: sensing and feeling,” *Neuron*, vol. 82, no. 4, pp. 737–755, 2014.
- [296] S. Haddadin, L. Johannsmeier, and F. D. Ledezma, “Tactile robots as a central embodiment of the tactile internet,” *Proceedings of the IEEE*, vol. 107, no. 2, pp. 471–487, 2018.
- [297] J. Liang, J. Wu, H. Huang, W. Xu, B. Li, and F. Xi, “Soft sensitive skin for safety control of a nursing robot using proximity and tactile sensors,” *Sensors Journal*, vol. 20, no. 7, pp. 3822–3830, 2019.
- [298] R. Kõiva, T. Schwank, G. Walck, M. Meier, R. Haschke, and H. Ritter, “Barometer-based tactile skin for anthropomorphic robot hand,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9821–9826.
- [299] R. Dahiya, N. Yogeswaran, F. Liu, L. Manjakkal, E. Burdet, V. Hayward, and H. Jörntell, “Large-area soft e-skin: The challenges beyond sensor designs,” *Proceedings of the IEEE*, vol. 107, no. 10, pp. 2016–2033, 2019.
- [300] E. Shahriari, A. Kramberger, A. Gams, A. Ude, and S. Haddadin, “Adapting to contacts: Energy tanks and task energy for passivity-based dynamic movement primitives,” in *Proc. International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 136–142.
- [301] M. Helmert, “The fast downward planning system,” *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.
- [302] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub, “Clingo= asp+ control: Preliminary report,” *arXiv preprint arXiv:1405.3694*, 2014.
- [303] J. Burmester, J. Dillinger, W. Escherich, E. Ignatowitz, S. Oesterle, L. Reißler, A. Stephan, R. Vetter, and F. Wieneke, *Fachkunde Metall*, 58th ed., ser. Europa Lehrmittel. Verlag Europa-Lehrmittel Nourney Vollmer GmbH & Co. KG, 2020.



- [304] H. Bumiller, M. Burgmaier, C. Duhr, W. Eichler, B. Feustel, T. Käppel, W. Klee, J. Manderla, O. Reichmann, J. Schwarz, K. Tkotz, and U. Winter, *Fachkunde Elektrotechnik*, 32nd ed., ser. Europa Fachbuchreihe für elektrotechnische Berufe. Verlag Europa-Lehrmittel Nourney Vollmer GmbH & Co. KG, 2021.
- [305] H. Hebel, W. Eichler, G. Lämmlin, C. Sartor, A. Scheib, P. Schott, O. Spielvogel, E. Thiele, and U. Winter, *Fachkunde Mechatronik*, 6th ed., ser. Europa-Fachbuchreihe für Mechatronik. Verlag Europa-Lehrmittel, 2020.
- [306] L. S. Homem de Mello and A. C. Sanderson, “A correct and complete algorithm for the generation of mechanical assembly sequences,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1989, pp. 56–61.
- [307] T. de Fazio and D. Whitney, “Simplified generation of all mechanical assembly sequences,” *Journal on Robotics and Automation*, vol. 3, no. 6, pp. 640–658, 1987.
- [308] Y. F. Huang and C. Lee, “Precedence knowledge in feature mating operation assembly planning,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 1989, pp. 216–221.
- [309] A. Koc, I. Sabuncuoglu, and E. Erel, “Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an and/or graph,” *IIE Transactions*, vol. 41, no. 10, pp. 866–881, 2009.
- [310] S. Russel and P. Norvig, *Artificial intelligence: a modern approach*. Pearson Education, Inc., 2002.
- [311] R. Dechter and J. Pearl, “Generalized best-first search strategies and the optimality of A\*,” *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 505–536, 1985.
- [312] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, “An experimental comparison of bayesian optimization for bipedal locomotion,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.
- [313] J. Snoek, “Bayesian optimization and semiparametric models with applications to assistive technology,” Ph.D. dissertation, University of Toronto, 2013.
- [314] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in Neural Information Processing Systems*, 2012, pp. 2951–2959.
- [315] R. M. Neal, “Slice sampling,” *Annals of Statistics*, pp. 705–741, 2003.
- [316] J. M. Hernández-Lobato, M. Gelbart, M. Hoffman, R. Adams, and Z. Ghahramani, “Predictive entropy search for bayesian optimization with unknown constraints,” in *Proc. International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 1699–1707.
- [317] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

- [318] N. Hansen, Ed., *Towards a New Evolutionary Computation*. Springer, Berlin, Heidelberg, 2006.
- [319] M. D. McKay, R. J. Beckman, and W. J. Conover, “Comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.
- [320] C. Daniel, G. Neumann, and J. Peters, “Hierarchical relative entropy policy search,” in *Proc. Artificial Intelligence and Statistics*, 2012, pp. 273–281.
- [321] F. Voigt, L. Johannsmeier, and S. Haddadin, “Multi-level structure vs. end-to-end-learning in high-performance tactile robotic manipulation,” in *Proc. Conference on Robot Learning (CoRL)*, 2021, pp. 2306–2316.
- [322] G. E. P. Box, W. H. Hunter, S. Hunter, *et al.*, *Statistics for experimenters*. John Wiley and Sons New York, 1978, vol. 664.
- [323] C. Nguyen, T. Hassner, M. Seeger, and C. Archambeau, “Leep: A new measure to evaluate transferability of learned representations,” in *Proc. International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 7294–7305.
- [324] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Amir R. Zamir, and Leonidas J. Guibas, “An information-theoretic metric of transferability for task transfer learning,” 2019.
- [325] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104.
- [326] M. J. Sorocky, S. Zhou, and A. P. Schoellig, “Experience selection using dynamics similarity for efficient multi-source transfer learning between robots,” in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2739–2745.
- [327] T. Kailath, “The divergence and bhattacharyya distance measures in signal selection,” *Transactions on Communications*, vol. 15, no. 1, pp. 52–60, 1967.
- [328] F. J. Massey Jr, “The kolmogorov-smirnov test for goodness of fit,” *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.
- [329] T. van Erven and P. Harremoës, “Rényi divergence and kullback-leibler divergence,” *Transactions on Information Theory*, vol. 60, no. 7, pp. 3797–3820, 2014.
- [330] S. Haddadin and L. Johannsmeier, “The art of manipulation: Learning to manipulate blindly,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.
- [331] J. Grischke, L. Johannsmeier, L. Eich, L. Griga, and S. Haddadin, “Dentronics: Towards robotics and artificial intelligence in dentistry,” *Dental Materials*, vol. 36, no. 6, pp. 765–778, 2020.

- [332] X. Chen, L. Johannsmeier, H. Sadeghian, E. Shahriari, M. Danneberg, A. Nicklas, F. Wu, G. Fettweis, and S. Haddadin, “On the communication channel in bilateral teleoperation: An experimental study for ethernet, wifi, lte and 5g,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 7712–7719.
- [333] K. Feldmann, Ed., *Handbuch Fügen, Handhaben, Montieren*, 1st ed., ser. Edition Handbuch der Fertigungstechnik. Hanser, 2014.
- [334] Deutsches Institut für Normung, “Fertigungsverfahren fügen - teil 1: Zusammensetzen; einordnung, unterteilung, begriffe,” 2003.
- [335] J. F. Broenink and M. L. J. Tierneho, “Peg-in-hole assembly using impedance control with a 6 dof robot,” in *Proc. European Simulation Symposium*, 1996, pp. 504–508.
- [336] S. R. Chhatpar and M. S. Branicky, “Search strategies for peg-in-hole assemblies with position uncertainty,” in *Proc. International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2001, pp. 1465–1470.
- [337] J. Dietrich and J. Dietrich, “Schneiden (zerteilen),” *Praxis der Umformtechnik: Umform-und Zerteilverfahren, Werkzeuge, Maschinen*, pp. 266–290, 2018.
- [338] R. Bogue, “Cutting robots: a review of technologies and applications,” *Industrial Robot: An International Journal*, vol. 35, no. 5, pp. 390–396, 2008.
- [339] EPIC Games, “Unreal engine,” 2022. [Online]. Available: <https://www.unrealengine.com/en-US/>
- [340] F. Pedregosa, G. Varoquaux, A. Gramfort, and V. Michel, “scikit-learn,” 2010. [Online]. Available: <https://scikit-learn.org/stable/index.html>
- [341] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, “Intel realsense stereoscopic depth cameras,” in *Proc. Conference on Computer Vision and Pattern Recognition*. IEEE, 2017, pp. 1–10.
- [342] C. Mönnink, L. Eich, S. Haddadin, M. Stiesch, and J. Grischke, “Dentronics: Tooth cleaning with a tactile, collaborative robot. an in vitro proof of concept.” *International Journal of Computerized Dentistry*, pp. 1–17, 2023.
- [343] M. Jayaweera, H. Amarasinghe, and N. W. Johnson, “Reshaping dental practice in the face of the covid-19 pandemic: Leapfrogging to dentronics,” *Oral Diseases*, 2021.
- [344] I. Spectrum. (2019) Dentronics: New application domain for collaborative robots in dental assistance. [Online]. Available: <https://www.youtube.com/watch?v=M8s9bS8qkRE>
- [345] A. Naceri, J. Elsner, M. Tröbinger, H. Sadeghian, L. Johannsmeier, F. Voigt, X. Chen, D. Macari, C. Jähne, M. Berlet, *et al.*, “Tactile robotic telemedicine for safe remote diagnostics in times of corona: System design, feasibility and usability study,” *Robotics and Automation Letters (R-AL)*, vol. 7, no. 4, pp. 10 296–10 303, 2022.

- [346] A. Moortgat-Pick, A. Adamczyk, T. Tomić, and S. Haddadin, “Feeling the true force in haptic telepresence for flying robots,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9789–9796.
- [347] P. der Moderne. (2021) Ki.robotik.design. [Online]. Available: <https://www.pinakothek-der-moderne.de/ausstellungen/ki-robotik-design/>
- [348] D. Zardykhan, P. Svarny, M. Hoffmann, E. Shahriari, and S. Haddadin, “Collision preventing phase-progress control for velocity adaptation in human-robot collaboration,” in *Proc. International Conference on Humanoid Robots (Humanoids)*. IEEE, 2019, pp. 266–273.
- [349] L. Johannsmeier and S. Haddadin, “Can we reach human expert programming performance? a tactile manipulation case study in learning time and task performance,” in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.
- [350] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.
- [351] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, “The computational limits of deep learning,” *arXiv preprint arXiv:2007.05558*, 2020.
- [352] A. Stemmer, G. Schreiber, K. Arbter, and A. Albu-Schäffer, “Robust assembly of complex shaped planar parts using vision and force,” in *Proc. International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, 2006, pp. 493–500.
- [353] E. Brochu, V. M. Cora, and N. de Freitas, “A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” *arXiv preprint arXiv:1012.2599*, 2010.
- [354] K. Swersky, J. Snoek, and R. P. Adams, “Multi-task bayesian optimization,” in *Advances in Neural Information Processing Systems*, 2013, pp. 2004–2012.
- [355] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.
- [356] W.-L. Loh, “On latin hypercube sampling,” *The Annals of Statistics*, vol. 24, no. 5, pp. 2058–2080, 1996.
- [357] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proc. International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 1861–1870.
- [358] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.
- [359] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

- [360] Francois-Michel de Rainville, “Deap,” 15.02.2021. [Online]. Available: <https://github.com/deap/deap>
- [361] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.
- [362] S. Haddadin, S. Haddadin, and S. Parusel, “Franka emika panda,” 04.02.2021. [Online]. Available: <https://www.franka.de/>
- [363] Y. Huang, M. Mahmudi, and M. Kallmann, “Planning humanlike actions in blending spaces,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011, pp. 2653–2659.
- [364] A. Billard and D. Kragic, “Trends and challenges in robot manipulation,” *Science*, vol. 364, no. 6446, 2019.
- [365] G. Michalos, S. Makris, L. Rentzos, and G. Chryssolouris, “Dynamic job rotation for workload balancing in human based assembly systems,” *CIRP Journal of Manufacturing Science and Technology*, vol. 2, no. 3, pp. 153–160, 2010.
- [366] D. Knobbe, H. Zwirnmann, M. Eckhoff, and S. Haddadin, “Core processes in intelligent robotic lab assistants: Flexible liquid handling,” in *Proc. International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2335–2342.
- [367] J. Ringwald, S. Schneider, L. Chen, D. Knobbe, L. Johansmeier, A. Swikir, and S. Haddadin, “Towards task-specific modular gripper fingers: Automatic production of fingertip mechanics,” *Robotics and Automation Letters (R-AL)*, vol. 8, no. 3, pp. 1866–1873, 2023.