Human-Inspired Compliant Controllers for Robotic Assembly

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Stefan Scherzinger

aus Freiburg i. Br.

Tag der mündlichen Prüfung: 09.07.2021
Erster Gutachter: Prof. Dr.-Ing. Rüdiger Dillmann
Zweiter Gutachter: Prof. Dr.-Ing. Alin Albu-Schäffer

Preface

Robotic automation is a key driver for the advancement of technology. Programmed by trained engineers and application programmers, industrial manipulators execute fast and repetitive tasks through highly precise instructions. While this is an established process for motion generation with little contact, force-sensitive assembly is still very challenging to program and automate. Based on constraint formulations, these recipes are strongly limited in their adaptivity to slight changes in positioning and unforeseen jamming. Human sensing and cognition, on the contrary, are powerful sources of inspiration but finding technical representations for these skills and translating them into robot controllers is part of ongoing research.

This work presents methods to extract and learn assembly skills from human performance and to transfer them into controllers on industrial robots. By combining model-based and data-driven approaches, the programming is reduced to robot-independent teleoperation of assembly tasks in a simulation environment. A special form of a recurrent neural network mimics this recorded behavior and learns force-sensitive strategies to handle part jamming and wedging for autonomous execution. This skill is then transferred to industrial robots with a novel, unifying force controller. The focus of this work lies on investigating technical realizations of such skills and their programming, and on the special requirements posed by compliant control for industrial robots. The developed methods are implemented in software and evaluated both in simulation and on real robotic manipulators. With this modeling and programming approach, this thesis hopes to contribute valuable insights towards the goal to endow robots with human skills for assembly.

Karlsruhe,
March 2021

Stefan Scherzinger
FZI Research Center for Information Technology

Zusammenfassung

in German

Automatisierung mit Robotern ist ein fester Bestandteil moderner Fertigungsprozesse und in vielen Industrienationen ein wichtiger Treiber von Wettbewerbsfähigkeit. Das Programmieren der Roboter erfordert dabei in der Regel Expertenwissen. In einem typischen Prozess werden Trajektorien offline erstellt und vor Ort an der Anlage nachjustiert. Das Ergebnis ist eine hochgenaue Beschreibung von speziellen Anweisungen mit stark eingeschränkter Anpassungsfähigkeit an kleinsten Positionsänderungen der Werkstücke. Besonders kraftsensitive Fügevorgänge sind jedoch sehr schwer zu programmieren und automatisieren. Durch Lokalisierungsfehler werden Unsicherheiten in die Prozessketten eingebracht, die in Verbindung mit dem geringen Spiel der Bauteile bereits bei geringen Fluchtungsfehlern zu Verklemmungen und Verkantungen führen. Das Lösen dieser Aufgaben kann nicht durch das bloße Abspielen von vorprogrammierten Bewegungen robust erreicht werden und benötigt daher häufig nach wie vor manuelle Montage. Die Übertragung von menschlichem Geschick in Roboterprogramme scheitert dabei vor allem an genauen Anweisungen zur Handhabung der Unsicherheiten, die wir eher intuitiv lösen: Bei manueller Montage lässt sich an der richtigen Stelle etwas *drücken, drehen oder rütteln*, diese Strategien sind jedoch technisch schwer zu quantifizieren und nachzubilden.

Die vorliegende Arbeit stellt einen hybriden Ansatz zwischen modellbasierten und daten-getriebenen Methoden vor, diese Strategien und Fähigkeit (engl. Skills) aus menschlichem Vormachen zu extrahieren und durch neu entwickelte Kraftregler auf Industrie-Roboter zu übertragen.

Der Fokus liegt auf Robotern mit gelenkseitiger Positions- oder Geschwindigkeitsregelung. Diese Systeme sind in industriellen Anlagen weit verbreitet und können über einen am Roboterflansch montierten Kraft-Momenten-Sensor durch eine äußere Regelungsschleife mit Nachgiebigkeit aufgerüstet werden. Eine kraftsensitive Interaktion ist in diesen Fällen erforderlich um große Reaktionskräfte zwischen Bauteil und Roboter zu vermeiden. Die Arbeit adressiert Anwendungsfälle, in denen die reale Hardware nicht für die Programmierung zur Verfügung steht. Systeme im Einsatz können damit durch ein Software-Update mit den neuen Fähigkeiten umgerüstet werden.

Die vorliegende Arbeit untersucht zunächst vorhandene Ansätze zur Implementierung von Nachgiebigkeitsregelung für die betrachteten Robotersysteme. Anschließend werden verwandte Arbeiten im Bereich des Skill-Learning in der Robotikforschung mit Fokus auf Montage-Applikationen analysiert und diskutiert. Eine qualitative Analyse bewertet die verschiedenen Ansätze unter aufgestellten Gütekriterien und motiviert die Kombination Modellbasierter mit datengetriebenen Methoden.

Die betrachteten Fügevorgänge werden in einem Simulator nach modelliert und über Teleoperation mit einem Joystick ausgeführt. Der Einsatz von Simulation stellt im Allgemeinen hohe Anforderungen an den Realismus physikalischer Effekte um den Trans-

fer der Lösungsansätze auf die reale Hardware zu garantieren. Die vorliegende Arbeit löst dieses Problem indem typische Verkantungen, die bei der späteren Ausführung mit dem Roboter auftreten können, mit erhöhten Reibwerten bewusst übertrieben werden. Physikalische Ungenauigkeiten im simulierten Kontaktverhalten der Bauteile werden genutzt, um die Montage auf künstliche Weise zu verkomplizieren und verlangen dadurch beim Vormachen und Lösen der Aufgabe mit dem Joystick robuste Strategien durch die Bediener.

Durch Aufzeichnung der kommandierten Steuerbefehle und der Relativbewegung der Bauteile werden Datensätze erhalten, die menschliche Skills implizit in das iterative Zusammenspiel zwischen eigenen Aktionen und beobachteter Auswirkung einbetten.

Basierend auf diesen Datensätzen wird die Modellierung der Skills anschließend als überwachtes Lern-Problem formuliert. Die Skills werden über spezielle Rekurrente Neuronale Netze modelliert, die reaktives Verhalten aus den aufgezeichneten Daten lernen und mit probabilistischen Komponenten die Ambivalenz des *Hin-und-her-Probierens* in engen Kontakten nachbilden können. Die Eingangs- und Ausgangsdaten der Skills werden dabei in roboterunabhängigen, objektbezogenen Koordinaten formuliert, wodurch die Skills frei im Arbeitsraum des Manipulators durch eine geeignete kartesische Regelung angewendet werden können.

Zum Transfer der Skills auf Roboter wird ein neuartiger Regelungsansatz entwickelt, der das Prinzip der Dynamiksimulation auf die Kraftregelung überträgt. Während der Ausführung generieren die gelernten Skill-Modelle vom Menschen inspirierte Steuersignale, die mit den Sollgrößen der Robotertrajektorien und den gemessenen Kontaktkräften des Sensors überlagert werden. Die resultierende Größe wird anschließend als Anregung für ein virtuelles Modell des Roboters verwendet, das die Bewegung für den echten Roboter vorsimuliert und ihn damit steuert. Über eine einfach anzuwendende Parametrisierung des virtuellen Systems kann ein lineares Verhalten im kartesischen Arbeitsraum des Reglers erreicht werden. Das Problem der Inversen Kinematik in singulären Gelenkkonfigurationen wird auf physikalisch plausible Weise durch den dynamikbasierten Regler gelöst und bietet damit eine stabile Alternative zu bestehenden Ansätzen.

Die entwickelten Theorien und Methoden werden in Experimenten in Simulation und auf Hardware evaluiert. Parameterstudien in Simulation untersuchen die Fähigkeit des vorgestellten Skill-Modells die Datensätze zu lernen und prüfen die Robustheit ihrer Ausführungen. Der neu entwickelte, dynamikbasierte Ansatz zur Lösung des Inversen Kinematik Problems wird gegen ausgewählte Referenzmethoden abgeglichen und mit dem neuen Regler auf verschiedenen Robotersystemen getestet. Ein abschließender Skill-Transfer von Simulation auf echte Hardware zeigt das Gesamtkonzept anhand zwei ausgewählter Anwendungen. Die Methoden und Beiträge der Arbeit werden im Folgenden im Detail erörtert.

Acknowledgments

This thesis was conducted during my work as a research scientist at FZI Forschungszentrum Informatik. Many people have accompanied me on this journey and have positively influenced this work.

First and foremost, I want to thank my supervisor, Prof. Rüdiger Dillmann, for giving me the chance to work at this research lab and for his continuous support throughout my thesis. He found the right balance between guiding me and giving me the freedom to develop my ideas and this provided a great working atmosphere. The many friendly and encouraging conversations we had and the fact that he understood the difficulty of balancing both project deadlines and our scientific goals took away much of my day-to-day stress. I also want to thank Prof. Albu-Schäffer for his interest in my work and for becoming my second supervisor.

Furthermore, I would like to thank our department manager Arne for the opportunities to work on many exciting projects, especially EuRoC and iBOSS, and for becoming a strong supporter of my ideas. Sharing them and collecting valuable feedback from so many colleagues has been a great opportunity and inspiration over the years. I will always have fond memories of our annual scientific retreats to Mallorca in the springtime and the engaging and fruitful discussions. Being surrounded by people that made me smile on so many occasions is why I enjoyed almost every day of work during my years at FZI. I am also indebted to the many bright individuals I have worked with on projects. It has been a privilege to be part of such an ambitious and hardworking team at IDS. With the right amount of motivation and humor, the endless hackathons, and even the sleepless nights for robot demos were fun. I am also grateful to the developers of the fantastic free and open-source software I built my research on, especially all the contributors to the ROS and Tensorflow frameworks and related libraries.

Following a goal over such a long period also requires emotional support: I am deeply grateful to my family, especially my parents Bruno and Gerlinde for always listening to me and caring about me and always believing in my abilities; my brother Johannes and my sister Julia for joyful family gatherings and tips for better English; and last but not least my wife Dina for her unlimited support and for being always there for me, ever since we met in the third semester of my undergraduate studies. For having my back and for reminding me in the right moments of my strengths. She is my biggest personal fan and motivator and the love of my life.

Contents

List of Figures

List of Tables

Notation and Acronyms

This section provides an overview about frequently used symbols and acronyms. We use bold uppercase to denote matrices, such as $\boldsymbol{J}, \boldsymbol{H}$. Exceptions are the vectors of Coriolis terms $\boldsymbol{C}$ and gravitational components $\boldsymbol{G}$. Vectors are depicted in bold lowercase, such as $\boldsymbol{f}, \boldsymbol{q}$, and are considered column vectors. Skalars have a lowercase normal font, e.g. α, with the exception of N and T that indicate index limits.

Notation

$\mathcal{A}_o$	Coordinate system with global orientation
$\mathcal{A}$	Coordinate system for center of mass
$\mathcal{B}$	Robot base frame
$\mathcal{E}$	Robot end-effector frame
$\mathcal{N}_c$	c-variate Gaussian
$\mathcal{O}$	Assembly origin, equals the goal pose of the active part
$\boldsymbol{C}$	Vector of Coriolis and centrifugal terms
$\boldsymbol{D}$	Damping matrix
$\boldsymbol{G}$	Vector of gravitational components
$\boldsymbol{H}$	Joint space inertia matrix
Ip_e	Virtual end-effector's polar moment of inertia
Ip_l	Virtual link's polar moment of inertia
$\boldsymbol{I}$	Identity matrix of suitable dimension
$\boldsymbol{J}$	Manipulator Jacobian
$\boldsymbol{K}_D$	Derivative gain matrix
$\boldsymbol{K}_P$	Proportional gain matrix
$\boldsymbol{K}$	Stiffness matrix
$\boldsymbol{M}$	Mass matrix
N	Sequential memory length of skill models
${}^{\text{to}}\boldsymbol{R}_{\text{from}}$	Rotation matrix
T	Number of data points in a demonstration
${}^{\text{to}}\boldsymbol{T}_{\text{from}}$	Homogeneous transformation matrix
$\boldsymbol{c}_t$	Cell state of the LSTM
c	Dimension of the final skill model output ($= 6$)
d_{lin}	Linear damping constant
d_{rot}	Rotary damping constant
$\boldsymbol{f}^c$	Control wrench
$\boldsymbol{f}^d$	Desired contact wrench
$\boldsymbol{f}^h$	Human motor command
$\boldsymbol{f}^n$	Netforce in Cartesian space
f_x, f_y, f_z	Force components

$\boldsymbol{f}$	Vector of measurements of a force-torque sensor
$\boldsymbol{h}_t$	Hidden state of the LSTM
k_d	Damping constant
k_p	Spring constant
k	Number of Gaussian kernels
m_e	Virtual end-effector mass
m_l	Virtual link mass
$\boldsymbol{q}$	Vector of joint positions
$\dot{\boldsymbol{q}}$	Vector of joint velocities
$\ddot{\boldsymbol{q}}$	Vector of joint accelerations
t_x, t_y, t_z	torque components
$\boldsymbol{x}$	Current end-effector pose
$\dot{\boldsymbol{x}}$	Current end-effector velocity
$\ddot{\boldsymbol{x}}$	Current end-effector acceleration
$\boldsymbol{x}^c$	Control signal
$\boldsymbol{x}^d$	Desired end-effector pose
$\dot{\boldsymbol{x}}^d$	Desired end-effector velocity
$\boldsymbol{x}^r$	Cartesian reference pose
$\dot{\boldsymbol{x}}^r$	Cartesian reference velocity
$\hat{\boldsymbol{x}}_t$	Input feature vector for skill models at time t
$\hat{\boldsymbol{y}}_t$	Label for predictions at time t
$\boldsymbol{z}$	Output vector of MDNs
$\boldsymbol{\Lambda}$	Joint space inertia matrix in operational space
$\boldsymbol{\Sigma}$	Scale matrix in SVD | covariance matrix
α	Damping term in DLS
$\boldsymbol{\alpha}$	Vector of mixing coefficients for MDNs
γ	Ratio of end-effector dominance
$\boldsymbol{\mu}$	Vector of means for multivariate Gaussian
ϕ_i	Kernel function
π	Policy
$\sigma_{\max}$	Maximal singular value
$\sigma_{\min}$	Minimal singular value
$\boldsymbol{\sigma}$	Vector of standard deviations for MDNs
$\boldsymbol{\theta}$	Vector of learnable parameters in skill models
$\boldsymbol{\tau}$	Vector of joint torques
$\boldsymbol{\omega}$	Angular velocity
ξ_i	Demonstrations

Acronyms

ABA	Articulated Body Algorithm
BC	Behavioral Cloning
BPTT	Backpropagation Through Time
BP	Backpropagation
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering

CDF	Cumulative Distribution Function
CRBA	Composite Rigid Body Algorithm
DLS	Damped Least Squares
DMP	Dynamic Movement Primitives
FDCC	Forward Dynamics Compliance Control
FD	Forward Dynamics
FK	Forward Kinematics
GMM	Gaussian Mixture Models
GMR	Gaussian Mixture Regression
HMM	Hidden Markov Models
IK	Inverse Kinematics
IL	Imitation Learning
JI	Jacobian Inverse
JT	Jacobian Transpose
LSE	Least Squares Estimate
LSTM	Long Short-Term Memory
LWR	Locally Weighted Regression
LfD	Learning from Demonstration
MDN	Mixture Density Network
PD	Proportional-Derivative
PDF	Probability Density Function
PI	Proportional-Integral
PbD	Programming by Demonstration
RCC	Remote Center Compliance
RL	Reinforcement Learning
RNEA	Recursive Newton-Euler Algorithm
RNN	Recurrent Neural Network
ROS	Robot Operating System
SDLS	Selectively Damped Least Squares
SVD	Singular Value Decomposition

1. Introduction

Robotic automation is a strongly growing field and an important driver for current and future manufacturing processes. Industrial manipulators are used for fast and repetitive tasks, they enable the handling of heavy and bulky workpieces, or they provide high precision whenever a perfectly steady hand is required. However, such manipulators are multi-purpose machines and need to be programmed for specific tasks. This requires trained engineers and programmers in offline and online cycles for planning and correcting trajectories. The result is a highly accurate description of instructions with strongly limited adaptivity to even the slightest changes in the workpieces' positions. Most often, experts are needed due to the complexity of the programming process, rather than due to the difficulty of the task itself: For instance, it is intuitive and simple to plug two connectors with our hands, even with closed eyes. However, describing the exact process quantitatively with sub-millimeter precise instructions for a robot program is comparatively hard. This is particularly challenging for force-sensitive assembly where a form of dexterity is required. In such instances, workers must use the right *feeling* and their experience. Finding technical representations for these skills and translating them into algorithms and robot controllers is an exciting and active field of research.

1.1. Motivation

Industrial robotics has come a long way from its inception and spans over half a century of research. Early applications of automation relied on force control and on users that teleoperated robotic manipulators through dangerous tasks remotely to mitigate the risk of injuries [1]. When in the late 60s and the early 70s computer programs started replacing human operators in these scenarios, they introduced the new problem of formulating force-motion strategies that humans had utilized naturally within teleoperation. Before that, human operators took control of task planning and force-motion response intuitively by using their vision and force sensing capabilities [1]. Since then, research has tried to find answers to this problem. Early work used neural networks to learn mappings from sensor measurements to corrective motion in the 90s [2], making first steps

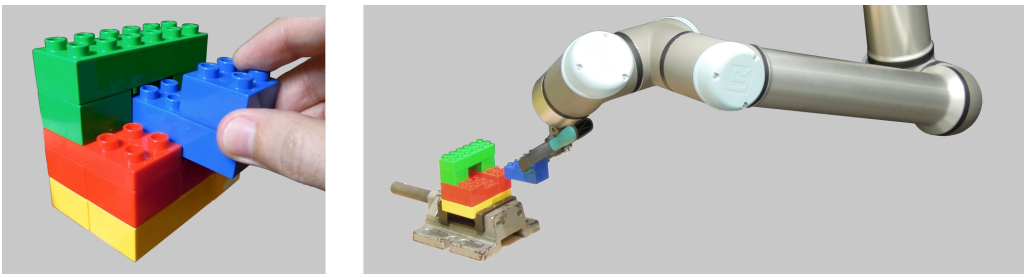


Figure 1.1.: Human-inspired compliant controllers for robotic assembly.

into that direction and predating many of the modern machine learning approaches by decades.

What makes assembly challenging is when the relative uncertainty between workpieces exceeds the task's clearance. Rigidly following pre-programmed trajectories is not feasible in these cases and contact forces must be anticipated to avoid damage to parts and robots. For only small discrepancies in positioning, the robot must apply some form of online correction during execution. Achieving this skill is crucial on the way to developing more autonomous systems and overcoming the bottleneck of carefully designed and calibrated environments in industrial settings. Humans are particularly good at handling these tasks. Using several senses, such as sight and touch, allows us to process a high bandwidth of information. In addition, we can draw on a huge amount of experience in object manipulation. Ever since childhood, humans interact with objects, toys first and later every kind of tool follows. Fig. 1.1 shows one such classic assembly. It is intuitive for us where to press, turn and jolt to get two parts mated. Jamming and wedging are dealt with almost unconsciously. These strategies are fuzzy and difficult to describe for compliant control. Thus, a key to achieving skill transfer to industrial robots lies within natural and intuitive programming methods so that the strategies we apply can manifest themselves through the method. Turning those capabilities into control strategies to enhance robotic assembly is open research. Much of the initial challenges remain for automation, and at the core of industrial assembly continues the incentive to endow manipulators with capabilities that come close to the skills of human workers.

1.2. Problem Statement and Research Questions

Embedded into the setting of offline programming [3], this thesis focuses on the specific problem of joining two parts through insertion, which, in the taxonomy of assembly falls into the category of *fine motion planning* [4]. Using this as the smallest operation, complex assemblies can be broken down into several such subtasks. The following descriptions clarify important problems and objectives.

Tight fits For the assembly tasks we consider, motion is highly constrained with low clearances between the parts. An important corner case includes plug connections where the parts tightly fit together. Compliant robot end-effectors are required in all cases to yield reaction forces and we need suitable force control algorithms to balance force and motion in contact. To scale to industrial use cases, our method should work with complex shapes and insertion directions and be independent of the dimensions of the workpieces.

Reactive skills Uncertainty is anticipated from perception in unstructured environments and complicates the previous assembly description: A localization error in combination with low clearances provokes intermediate part jamming and can cause the robots to fail drastically even for infinitesimal misalignments. We seek corrective behavior during execution that is inspired by human skill to enable more autonomous systems.

Offline programming Task programming without access to the real hardware poses a substantial challenge for insertion assembly. Contacts and tight fits are difficult to simulate realistically, so that using simulation creates the additional challenge of successfully

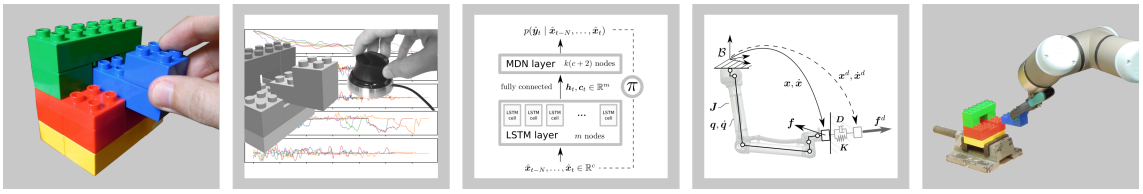


Figure 1.2.: Concept overview: From human skills to compliant robot control. Demonstrators perform the assembly tasks in simulation environments and generate examples of error-correcting behavior. A special type of neural network learns strategies from these data, which can be executed through a unifying force controller on robotic manipulators.

transferring results to real hardware. Furthermore, the formulation of the assembly skills should be robot-independent to deploy them on different industrial manipulators, for which suitable robot controllers must provide the interfaces.

Research Questions

Concluding from the problems and objectives, the following research questions shall lead through the thesis:

Q-I *How to design a simple and intuitive method to program robotic skills for force-sensitive assembly tasks offline?*

Q-II *How to extract and incorporate human skill and intuition to make robots handle part jamming and wedging autonomously during execution?*

Q-III *How to implement controllers to transfer these offline-programmed skills to industrial manipulators?*

1.3. Concept Overview

This thesis takes a hybrid approach between model-based and data-driven components. The overall concept is to formulate robot-independent assembly skills based on simulated data with techniques from machine learning and to deploy these models through unifying robot controllers on the real systems. This separation of skill and robot control enables us to program offline, but puts the challenge of simulation and transfer into the foreground. Figure 1.2 shows an overview.

At the beginning of this process, there is an assembly task that we wish to program and automate from a manual baseline. The concept uses toy bricks for illustration. For practical use cases, the assembly task can comprise many challenges from the previous section with complex shapes, low clearances during insertion and tight fits at the end. The intuitiveness of programming plays a decisive role in capturing and using human skill. To this end, our concept incorporates the Programming by Demonstration paradigm (PbD) [5] by showing the assembly in a simulator with a joystick-like teach-device. This part of the concept is model-based and we replicate the task with simulating contact physics and friction using simplified meshes of the workpieces.

Through recording discrete state space and steering commands during demonstrations, we obtain a dataset that comprises human behavior for this assembly task. Repetitive demonstrations shall collect corrective strategies for challenging configurations. The idea is to provoke part misalignments in the simulator that are likely to happen during the real assembly and thus obtain a dataset with valuable solutions to learn from.

The next part of the concept is data-driven and consists of approximating this dataset with machine learning approaches. We use a combined form of neural network for learning patterns in sequential-probabilistic data. This model must be capable of capturing the ambiguity of strategies with much *trial and error* in tasks with low part clearance. The goal is to obtain models that predict error-reducing strategies for robot controllers.

Up to this point, the skills are robot-independently formulated with respect to object-relative coordinate frames. This makes them applicable everywhere in the manipulator's workspace through a robot controller that operates in the Cartesian regime. Industrial robots possess inner control loops, which provide limited interfaces for deploying these skills directly. Our concept solves this problem by proposing a unifying controller that relies on dynamics-based simulations and introduces a new way of formulating control loops for industrial robots. During execution on the real system, the skill guides the robot during assembly and can handle unforeseen part jamming and wedging with strategies inspired by human skill.

1.4. Contribution

This thesis proposes a new method for offline programming human-inspired assembly skills for industrial robots. The primary contributions of this research are:

- A comprehensive literature survey on skill programming and skill learning in robotics research with a focus on robotic assembly. This collection provides both historic and latest achievements to give an adequate overview of this diverse field.

- An approach to capture and model human-inspired assembly skills from demonstrations in simulation. The proposed methods create sequential-probabilistic, robot-independent models that let robots deploy error-correcting strategies against unforeseen part jamming. They can be used in industrial settings to gain more autonomy in unstructured environments.

- A unifying compliant controller tailored for industrial manipulators to execute assembly skills. The controller builds on simplified dynamics simulations and provides a new solution to the inverse kinematics problem for the field of manipulator control.

Accompanying analysis and experiments investigate and validate the new concepts. The contributions shall deepen our understanding of human assembly strategies, provide a possible technical representation, and motivate the usage of simulations both for demonstration and active compliant control. The implementation of the algorithm for robot control is available as free and open-source software[1].

[1] https://github.com/fzi-forschungszentrum-informatik/cartesian_controllers

1.5. Document Outline

Chapter 2 builds the theoretical background for this thesis on compliant control for industrial manipulators. Related work is then reviewed with a focus on robotic assembly and associated skill learning in robotics. The chapter closes with a qualitative analysis of the different methods and open potential.

Chapter 3 introduces the concept of human-inspired skills and formulates modeling them as a supervised learning problem. Assumptions and own definitions are described along with the simulation environment. The core of this chapter is the recording and modeling of skills with neural networks. Decisions about network architecture are based on prior data analysis of exemplary demonstrations. The chapter closes with an interface to robot control.

Chapter 4 introduces a new simulation-based control paradigm that leads to a new Cartesian compliant controller for industrial manipulators. A focus lies on achieving linearity in operational space and the combination with assembly skills as a special case of application.

Chapter 5 provides an in-depth evaluation of the central propositions of the thesis. The learning of skills and their performance is considered within parameter studies. Experiments both in simulation and on real robots evaluate the new controller design. The chapter closes with proof-of-concept demonstrations for two assembly use cases.

Chapter 6 summarizes the results by re-iterating the research questions and reconsidering the proposed methods under initial criteria. It highlights achievements, general implications and gives suggestions for further research.

2. Theoretic Background and Related Work

At the core of robotic assembly lies force-sensitive interaction. Our focussed industrial manipulators do not possess this feat by default, and we first collect important theoretic background on compliant control in Section 2.1. Spanning several decades of research, a contribution to the field of robotic assembly requires a good knowledge of the different approaches and associated methods to obtain skills. Section 2.2, therefore, makes a broad collection that includes both historic and latest achievements. Basing on this overview, Section 2.3 analyses the advantages and disadvantages of current methods under the problem statement from Section 1.2 and identifies the open potential that we target with our contribution.

2.1. Compliant Robot Control

In this thesis, we consider motion-controlled, industrial robots. Their strengths are fast trajectory execution with high path accuracy and end-effector loads. Leveraging high-gain position control in their joints, modern variants usually possess repeatable end-effector accuracy in the range of $0.1\,\mathrm{mm}$ to $0.01\,\mathrm{mm}$, which in theory gives them a far more *steady hand* when controlling any tool than human workers. Yet, leveraging these robots for highly constrained tasks, such as insertion assembly poses an important requirement: Even infinitesimal misalignments can, together with the high material stiffness of the work pieces involved, build up high reaction forces and damage parts and robots. On the way to more system autonomy, visual perception in unstructured environments replaces part-specialized fixtures and introduces uncertainty into the process chain. Assuring safe interaction despite this uncertainty makes compliantly suspended end-effectors mandatory. There are two dominant approaches of achieving this behavior: Passive, compliant devices [6] that go back to Whitney's Remote Center Compliance (RCC) [7], and active compliance by control, whose name was coined by Mason in the early 80s [8]. Both still being important fields of robotics research, this thesis contributes to the latter with a new compliant controller (Chapter 4).

To have a sufficient basis for these developments, Section 2.1.1 first lists important requirements for the industrial manipulators considered in this thesis. Section 2.1.2 and Section 2.1.3 present current control schemes that are suitable to realize active compliance by control on these robots. And Section 2.1.4 finally reviews solutions to the Inverse Kinematics (IK) problem that is at the core of these control schemes.

2.1.1. System Requirements

Active Compliance

Continuous force control started in the early 70s by Groome [9], and Nevins and Whitney [10]. The term *active compliance* was later coined by Mason [8] and refers to the mimicking of passive elements through closed-loop force control. Fig. 2.1(a) shows a

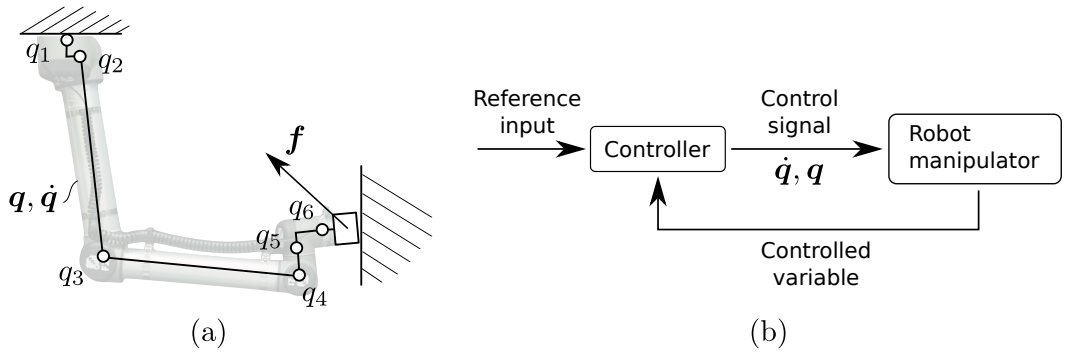


Figure 2.1.: Interaction with compliance. (a) Schematic illustration of the robots considered in this thesis: Six-axis, revolute joints with wrist force-torque sensor. (b) General control loop for realizing active compliance via joint position/velocity interfaces. We refer to these systems as *motion-actuated* systems.

schematic manipulator in contact with the environment. A wrist-mounted sensor measures the six Cartesian axes of forces and torques $\boldsymbol{f}$ that is actively used to compute joint positions $\boldsymbol{q} = (q_1, \ldots, q_6)$ and velocities $\dot{\boldsymbol{q}}$ in reaction to those signals. Other than lightweight robots [11], [12], [13] that use torque sensors in their joints, this add-on is an effective way of enhancing a big amount of existing robots with compliant behavior.

Since active compliance computes reaction after $\boldsymbol{f}$ occurs, they are inherently bound to delays and deadtimes of signal processing. Passive devices thus have advantages over control approaches in terms of physical shock absorption and energy dissipation, which are impossible to mimic by control on non-backdrivable systems. On position-controlled systems, this gives rise to the inherent problem of contact stability [14]. Hybrid devices exist and measure force-torque signals under the macro deformation of flexible sensors [15]. This, however, can benefit oscillations and requires the calibration of passive compliance to enable fine manipulation. Identification and usage of the manipulators' structural compliance can further improve active force control [16]. The strong points of active compliance and the reason for using this approach in this thesis is the flexibility of compliance shaping: Especially for motion that is sufficiently slow in contact, the inclusion of measurements from a wrist force-torque sensor allows to render arbitrary non-linear behavior that can easily be parameterized for specific tasks. For this thesis, this approach provides a control interface for complex assembly strategies that use both motion and force-torque setpoints.

Position/Velocity Joint Control Interfaces

In recent years, the increasingly popular Robot Operating System (ROS) [17] has emerged as a powerful software framework to support robotics research through a rich set of tools and algorithms. Its industrial-oriented initiative ROS-Industrial[1] has brought software control interfaces to industrial robot drivers[2], among them KUKA, ABB, Motoman, Universal Robots, and has connected robotics research to the industrial sector. This thesis focuses on compliant controllers for these systems, supporting a big amount of existing robots.

[1] https://rosindustrial.org/, accessed 3.11.2020
[2] https://github.com/ros-industrial, accessed 3.11.2020

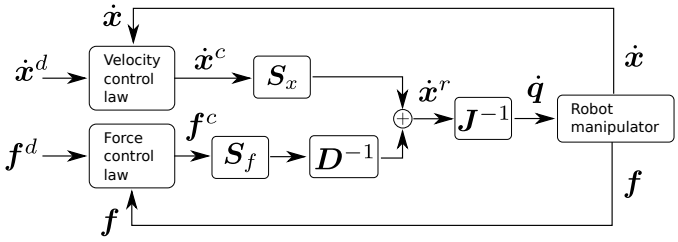


Figure 2.2.: Hybrid force/position control on motion-actuated systems.

Research on compliant manipulator control over the last decades, however, often assumes torque-based control interfaces. They require modifications of the industrial robot controllers but allow for linearization of the robot dynamics that leads to better error correction in the control laws. This was initially proposed by Shin and Lee [18] who added force control to pure motion control [19] in the hybrid framework of Raibert and Craig [20]. The term *acceleration-resolved* [19] refers to formulating the control problem based on acceleration terms at the robot hand, meaning that the highest order of error rejection in the control law is on the acceleration level [21]. Gravity, Coriolis, and Centrifugal terms sum up with the inertia-multiplied control signal to a net force $\boldsymbol{f}^n$ in Cartesian space, which is then mapped to joint torques $\boldsymbol{\tau}$ with

$$\boldsymbol{\tau} = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f}^n \tag{2.1}$$

under the usage of the manipulator's Jacobian transpose $\boldsymbol{J}^T$. This is also at the core of Khatib's operational space formulation [22], [23] which is still widely adopted to derive robot controllers for compliant interaction.

The challenge for the industrial manipulators considered in this thesis, however, is that they do not expose $\boldsymbol{\tau}$-based control interfaces. Instead, control engineers are limited to *velocity-resolved* approaches [24] and fall back to $\boldsymbol{q}, \dot{\boldsymbol{q}}$-based control interfaces for the joints. Fig. 2.1(b) shows a generic control scheme for these systems, in which the robot manipulator is treated as a black box. In further sections, we will refer to these systems as *motion-actuated* systems to underline the available interfaces for controller design. Villani and De Schutter also denote them as simplified systems [24], because they wrap the complexity of an inner, dynamics-linearizing control loop from external access. Not having access on torque level, the Jacobian inverse $\boldsymbol{J}^{-1}$ then maps a computed Cartesian reference velocity $\dot{\boldsymbol{x}}^r$ to joint space with

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}}^r. \tag{2.2}$$

Targeting motion-actuated robots requires a focused review[3] of control approaches that leverage Eq. (2.2) at the core of their control loops.

2.1.2. Hybrid Force/Position Control

For hybrid force/position approaches, a characteristic feature is the usage of two mutually different subspaces for separate force and position control. The concept was initially proposed for torque-actuated robots in the work of Craig and Raibert [27], [20] and is also

[3] More general reviews of the whole field of force and compliant control are provided by Villani et al [24], Calanca et al [25] and Schumacher et al [26].

credited to Mason's work [8]. Here we describe the version for motion-actuated systems in an adapted notation from [24].

The approach computes two separate control signals $\boldsymbol{x}^c$, $\boldsymbol{f}^c$ based on the error between desired end-effector velocity $\dot{\boldsymbol{x}}^d$, desired contact wrench $\boldsymbol{f}^d$ of the task, and current measurements $\dot{\boldsymbol{x}}$, $\boldsymbol{f}$, respectively. Both proportional (P) and proportional-integral (PI) gains are commonly used, e.g. [24]

$$\begin{aligned} \dot{\boldsymbol{x}}^c &= \dot{\boldsymbol{x}}^d + \boldsymbol{K}_{Ix} \int_0^t (\dot{\boldsymbol{x}}^d(\tau) - \dot{\boldsymbol{x}}(\tau)) d\tau \\ \boldsymbol{f}^c &= \boldsymbol{f}^d + \boldsymbol{K}_{Pf}(\boldsymbol{f}^d - \boldsymbol{f}) + \boldsymbol{K}_{If} \int_0^t (\boldsymbol{f}^d(\tau) - \boldsymbol{f}(\tau)) d\tau, \end{aligned} \tag{2.3}$$

with positive semi-definite gain matrices $\boldsymbol{K}_{Ix}$ for motion and $\boldsymbol{K}_{Pf}, \boldsymbol{K}_{If}$ for force control. The integral gain eliminates steady state errors for setpoint tracking. Simpler velocity and force controllers can be used and depend on the task's objective.

Two selection matrices $\boldsymbol{S}_x$ and $\boldsymbol{S}_f$ merge these control signals into a common reference velocity $\dot{\boldsymbol{x}}^r$ according to

$$\dot{\boldsymbol{x}}^r = \boldsymbol{S}_x \dot{\boldsymbol{x}}^c + \boldsymbol{D}^{-1} \boldsymbol{S}_f \boldsymbol{f}^c, \tag{2.4}$$

in which an inverse damping matrix $\boldsymbol{D}^{-1}$ maps force quantities to velocity space. Fig. 2.2 shows the closed-loop control scheme that uses Eq. (2.2) for joint actuation. Characteristic for the hybrid approach, the selection matrices satisfy $\boldsymbol{S}_x^T \boldsymbol{S}_f = \boldsymbol{0}$ in each configuration and users specify mutually exclusive components of $\boldsymbol{f}^d$ and $\dot{\boldsymbol{x}}^d$ for their tasks.

Parallel Force/Position Control

A slightly different version drops the selection matrices of the hybrid approach and is referred to as parallel force/position control [28], [29], [30]. We use the version of Villani et al [24] in our notation. One of the drawbacks of the previous approach is that specifying both $\boldsymbol{S}_x$, $\boldsymbol{S}_f$, and setting force and motion setpoints on different axes requires geometric knowledge about the environment. This becomes especially evolved for changing frames [24]. The parallel approach circumvents this by overlapping both target force and target motion in all of the six Cartesian dimensions so that Eq. (2.4) simplifies to

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})(\dot{\boldsymbol{x}}^c + \boldsymbol{D}^{-1} \boldsymbol{f}^c). \tag{2.5}$$

The control signals $\dot{\boldsymbol{x}}^c$ and $\boldsymbol{f}^c$ are those from Eq. (2.3). In a slight variation, Chiaverini et al used PD gains for the position controller and PI gains for the force controller to preference the elimination of force errors for the steady state [29]. Setting $\dot{\boldsymbol{x}}^c \equiv \boldsymbol{0}$ achieves a pure force controller on motion-actuated systems.

2.1.3. Admittance Control

In contrast to the hybrid and parallel approaches, admittance control [31] along with impedance control [32] model the end-effector's behavior with a dynamical system. A clear distinction is drawn by Hogan's work on interaction [32], which illustrates an impedance as a conversion from *flow to effort* and an admittance as a conversion from *effort to flow* [32]. In accordance, Ott et al [33] describe impedance control as closing motion control around inner force control loops, and admittance control as closing force control

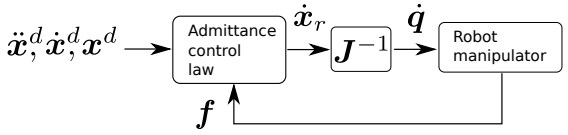


Figure 2.3.: Admittance control with motion-actuated systems.

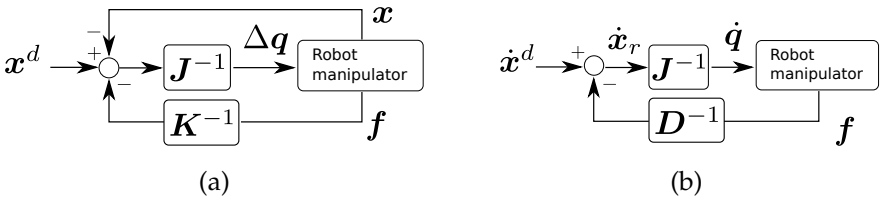


Figure 2.4.: Two simplifications of admittance control that do not require numerical integration: (a) Compliance control and (b) damping control.

around inner motion control loops, respectively. Consider a general spring-damper-mass relationship as end-effector model with

$$\boldsymbol{M}\Delta\ddot{\boldsymbol{x}} + \boldsymbol{D}\Delta\dot{\boldsymbol{x}} + \boldsymbol{K}\Delta\boldsymbol{x} = \boldsymbol{f}, \quad \Delta\boldsymbol{x} = \boldsymbol{x}^d - \boldsymbol{x}\ . \tag{2.6}$$

Impedance control substitutes the time derivatives of the desired trajectory $\boldsymbol{x}^d(t)$ and the current state $\boldsymbol{x}(t)$ into Eq. (2.6) to obtain $\boldsymbol{f}$, which actuates robot joints with Eq. (2.1). Admittance control on the other hand treats Eq. (2.6) as a differential equation of the variable $\boldsymbol{x}(t)$. The desired trajectory $\boldsymbol{x}^d, \dot{\boldsymbol{x}}^d, \ddot{\boldsymbol{x}}^d$ in this case takes the role of the system's rest position and $\Delta\boldsymbol{x} := \boldsymbol{x} - \boldsymbol{x}^d$. The solution $\boldsymbol{x}(t)$ can be found by solving Eq. (2.6) as an initial value problem through numerical integration. Using e.g. the forward Euler method requires transforming Eq. (2.6) into a set of 1st order ODEs in state-space representation. The system's motion is computed iteratively from initial conditions, using the control cycle's duration as step width. This leads to $(\boldsymbol{x}\ \dot{\boldsymbol{x}})^T$ in state-space representation, of which the simulated $\dot{\boldsymbol{x}}$ is taken as Cartesian reference velocity $\dot{\boldsymbol{x}}^r$ and mapped with Eq. (2.2) to joint space. Fig. 2.3 shows the closed-loop scheme of a general admittance controller.

Compliance Control

This simplification goes back to the work of Salisbury [34] and is today considered as a special case of admittance control without first and second-order dynamics. Error rejection is proportional to linear and angular displacement [24], leading to

$$\Delta\boldsymbol{q} = \boldsymbol{J}^{-1}(\boldsymbol{q})(\Delta\boldsymbol{x} - \boldsymbol{K}^{-1}\boldsymbol{f}). \tag{2.7}$$

Fig. 2.4(a) shows the closed-loop scheme. To be consistent with quantities, joint actuation is incremental in this scheme with $\boldsymbol{q} + \Delta\boldsymbol{q}$ in each control cycle.

Damping Control

Also being considered an admittance control type today, damping control goes back to the early work of Whitney [35]. Its control loop models the relationship between end-effector contact wrench and desired velocity [24]. On motion-actuated systems, setting

$\Delta \boldsymbol{x}, \Delta \ddot{\boldsymbol{x}} \equiv \mathbf{0}$ and $\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}^r$ simplifies Eq. (2.6) with Eq. (2.2) to

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})(\dot{\boldsymbol{x}}^d - \boldsymbol{D}^{-1}\boldsymbol{f}). \tag{2.8}$$

Fig. 2.4(b) shows the closed-loop scheme. Similar to Fig. 2.3, damping control does not include motion feedback.

2.1.4. Inverse Kinematics for Compliant Control

In admittance control, numerically integrating Eq. (2.6) can be interpreted as a forward dynamics simulation, in which $\boldsymbol{x}, \dot{\boldsymbol{x}}$ form the result in each control step. In contrast to the other compliant control schemes, having both as reference signals offers two options for joint actuation: The first one is the classic problem of Inverse Kinematics (IK) and refers to finding joint positions $\boldsymbol{q}$ for the target pose $\boldsymbol{x}^r$ of the manipulator's end-effector. Closed-form solutions are among the fastest and can be derived analytically for specific manipulators [36], [37] [4]. Hybrids between sample-based and gradient-based methods excel at finding solutions for highly redundant systems [38].

The second option and the one used for the other velocity-resolved control schemes is mapping end-effector velocity to joint space with $\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{x}}^r$. However, inverting the manipulator's end-effector Jacobian in each control step requires a square, full-rank matrix.

Redundancy and Singularities

The Jacobian of redundant manipulators is non-square. For these cases, Singular Value Decomposition (SVD) can compute the Moore-Penrose inverse $\boldsymbol{J}^+$, which minimizes the squared Euclidean norm $\|\boldsymbol{J}\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}^r\|^2$ [39]. First, the manipulator Jacobian is factorized according to $\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, in which $\boldsymbol{U}$ and $\boldsymbol{V}^T$ are orthogonal matrices. The diagonal matrix $\boldsymbol{\Sigma}$ determines the scale of the mapping from Cartesian to joint space. Its entries $\sigma_{ii} \geq 0$ are referred to as singular values. The pseudoinverse is then composed with

$$\boldsymbol{J}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^T. \tag{2.9}$$

The entries of $\boldsymbol{\Sigma}^+$ are the non-zero, inverted singular values $\sigma_{ii}^+ = 1/\sigma_{ii}$. Near singular configurations, however, $\boldsymbol{J}$ becoming rank-deficient results in vanishing σ_{ii} and conversely exploding σ_{ii}^+ for $\boldsymbol{J}^+$. This effect causes numerical instability also in the non-redundant inverse $\boldsymbol{J}^{-1}$. In both cases, even small $\dot{\boldsymbol{x}}^r$ cause huge joint speeds $\dot{\boldsymbol{q}}$ in the control loops.

Manipulability and Stability

An effect that strongly correlates with singular configurations is the loss of manipulability, for which Yoshikawa's measure defines $\sqrt{\det(\boldsymbol{J}\boldsymbol{J}^T)}$ for redundant, and $|\det(\boldsymbol{J})|$ for non-redundant systems [40]. The manipulator is unable to move instantaneously in all directions, which is also visible via σ_{min} as the smallest singular value of the mapping matrix from task space to joint space [41]. In combination with the biggest value σ_{max}, both singular values offer insights on the scaling behavior: When σ_{min} becomes zero, at least

[4] A popular software framework is *ikfast*: http://openrave.org/docs/0.8.2/openravepy/ikfast/, accessed 6.11.2020

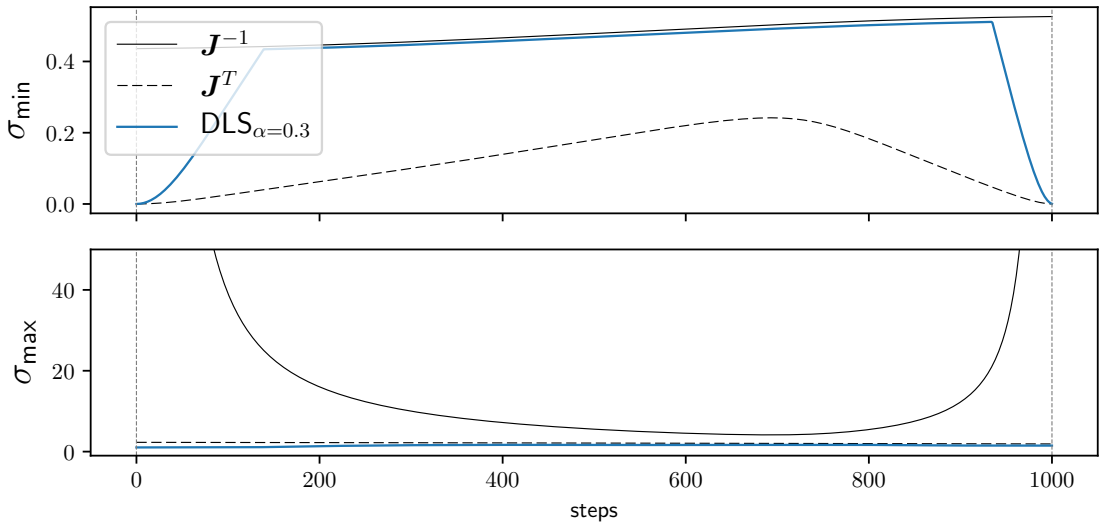


Figure 2.5.: Manipulability and stability near singularities. To obtain the plots, we interpolated between two singular configurations, indicated with dashed, vertical lines, and computed σ_{min} and σ_{max} of the mapping matrices $\boldsymbol{J}^T$,$\boldsymbol{J}^{-1}$ and DLS for comparison.

one component of the task space reference velocity $\dot{\boldsymbol{x}}^r$ gets lost in joint space. In contrast, σ_{max} indicates the most sensitive component concerning the impact on joint velocity $\dot{\boldsymbol{q}}$. Fig. 2.5 shows both values for different IK approaches near singular configurations. We used the kinematics of the *Universal Robots UR10*[5] for this illustration. The Jacobian transpose $\boldsymbol{J}^T$ stays stable but loses manipulability with σ_{min} dropping to zero. $\boldsymbol{J}^{-1}$ maintains high manipulability but scales infinitely with σ_{max} towards singularities. A remedy is limiting high joint velocities $\dot{\boldsymbol{q}}$ at the expense of degraded accuracy in Cartesian space.

Damped Least Squares (DLS)

The Damped Least Squares (DLS) method is an application of the Levenberg-Marquardt stabilization to manipulator control [42], [43]. It tries to circumvent instabilities of $\boldsymbol{J}^{-1}$ near singular configurations by adding a penalty term against excessive joint speeds to the minimization of the pseudoinverse approach:

$$\|\boldsymbol{J}\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}^r\|^2 + \alpha^2\|\dot{\boldsymbol{q}}\|^2. \tag{2.10}$$

Using the corresponding *normal equation*, the solution that minimizes this quantity is given by [43], [39]

$$\dot{\boldsymbol{q}} = (\boldsymbol{J}^T\boldsymbol{J} + \alpha^2\boldsymbol{I})^{-1}\boldsymbol{J}^T\dot{\boldsymbol{x}}^r. \tag{2.11}$$

With the damping term $\alpha \neq 0$, the matrix $(\boldsymbol{J}^T\boldsymbol{J} + \alpha^2\boldsymbol{I})$ is of full rank and thus invertible. Fig. 2.5 shows how the DLS aims for a trade-off between manipulability and stability. By setting $\alpha = 0$ as a lower limit, Eq. (2.11) falls back to the pseudoinverse approach.

A popular enhancement to DLS is the Selectively Damped Least Squares (SDLS) [44]. The method converges faster and circumvents to choose task-specific damping parameters α by adding σ_{ii}-specific damping terms to $\boldsymbol{\Sigma}$. Other methods include the more recent

[5] https://www.universal-robots.com/products/ur10-robot/, accessed 4.12.2020

Exponentially Damped Least Squares (EDLS) [45] with easy user-specific parameterization and a focus on avoiding typical singular constellations in human-robot interaction.

2.1.5. Discussion

Uncertainty in unstructured environments requires some sort of compliance for safe interactions between workpieces. Active compliance by control offers greater flexibility in parameterizing suitable behavior over passive devices. In this setting, a wrist-mounted force-torque sensor allows for designing task space controllers with force-torque and motion setpoint tracking. Many robots in industrial applications can easily be equipped with such sensors for active control, but at the same time do not provide access to the inner torque control loops of their actuators. By focusing on these *motion-actuated* systems, we include a big field of robots, but also an additional challenge: Acceleration-resolved methods that access inner torque-controlled loops are not applicable. Available control schemes are thus mainly velocity-resolved approaches that compute Cartesian reference velocities, which ultimately get mapped to joint space with the Jacobian inverse. Using this inverse as a central element poses stability issues near singular configurations. Established methods thus find trade-offs between accuracy and stability.

Having focused on motion-actuated robots and available methods for control, the next Section 2.2 investigates approaches for programming robots in assembly tasks. We open this investigation beyond motion-actuated systems to sufficiently reflect the broad field of available research.

2.2. Robotic Assembly

We first illustrate a simple assembly setting in Section 2.2.1 that introduces common problems and builds a baseline for the investigation of related work. We then review existing methods for deriving assembly skills. While not all methods specifically target assembly, they are established for contact-dominated tasks within the wider field of robotic manipulation research and merit consideration. A clear categorization is often difficult because many approaches use various methods jointly. We nevertheless attempt a clustering according to what we believe is at the core of each approach. The following Section 2.2.2 to Section 2.2.7 group the methods into categories, roughly ordered according to how much engineering thinking is required every time a new task is to be programmed: Section 2.2.2 starts with analytical methods that mainly rely on engineering skill and primitives. Section 2.2.3 presents methods that base their task description on the formulation of constraints. Section 2.2.4 discusses works that tackle assembly within the field of motion planning. Section 2.2.5 and Section 2.2.6 attempt to parameterize probabilistic and dynamic models directly from human performance. And finally, Section 2.2.7 presents methods that use algorithms to obtain assembly skills on their own through learning from trial and error.

2.2.1. An Introductory Example

In the scenario of assembly, we consider one of the parts to be *passive*, i.e. fixed with the environment, or held in place with another manipulator, and given with some uncertainty. The other part is *active*, i.e. grasped with a robotic gripper and controlled

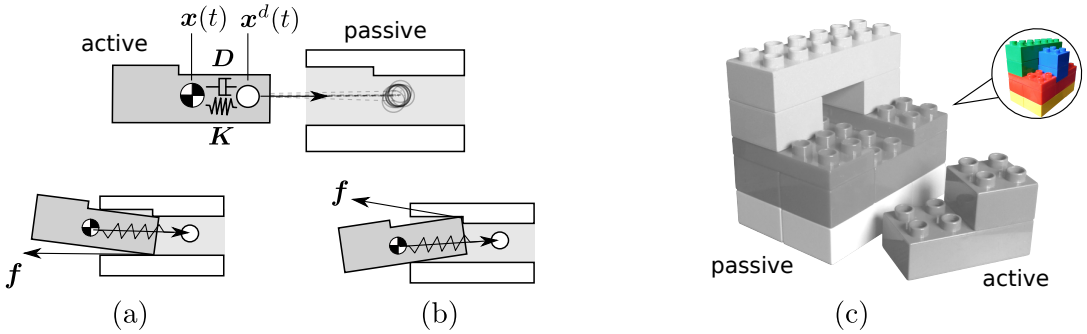


Figure 2.6.: Two-part assembly examples. Illustration of force closure (a) and form closure (b) that frequently arise in peg-in-hole problems with uncertainty. And a toy problem (c) that is hard for robotics due to additional insertion directions.

as the robot's end-effector. What makes part assembly challenging is when the relative uncertainty exceeds the task's clearance, which occurs in modern settings for gaining autonomy through the use of visual perception systems in unstructured environments.

Fig. 2.6 shows two assemblies that become challenging in such settings: The peg-in-hole problem on the left is iconic in assembly literature. Slightly varying in shapes and sizes, this type is often considered a benchmark for applied methods. The second on the right is a toy that some of us might be familiar with from our childhood days. For robotics, this is a challenging problem and adds two-step insertion directions, low tolerances and fits, and higher geometric complexity that make reasoning difficult against uncertainty.

During playback of the target motion $\boldsymbol{x}^d(t)$ on a real system, some form of end-effector compliance needs to consider environmental uncertainties. In the scheme of admittance control from Section 2.1.3, the relationship between target motion $\boldsymbol{x}^d$, actual motion $\boldsymbol{x}$, and external forces $\boldsymbol{f}$ can be described with a first-order relationship:

$$\begin{aligned} \boldsymbol{f} &= \boldsymbol{K}\Delta\boldsymbol{x} + \boldsymbol{D}\Delta\dot{\boldsymbol{x}} \\ \Delta\boldsymbol{x} &= \boldsymbol{x}^d(t) - \boldsymbol{x}(t) \end{aligned} \tag{2.12}$$

where $\boldsymbol{K}$ and $\boldsymbol{D}$ are user-specified stiffness and damping of the end-effector. In steady state during replay, contact forces alter the target trajectory by $\boldsymbol{f}\boldsymbol{K}^{-1} = \Delta\boldsymbol{x}$. For simple geometry, $\boldsymbol{K}^{-1}$ takes the role of an *accommodation matrix* [46] (admittance) that can correct part misalignments. Depending on the complexity of insertion, however, this is often not the case. There are two challenging phenomena. The first one is force closure, Fig. 2.6 (a), in which suitably parameterizing $\boldsymbol{K}$ might align the parts. The second challenge is form closure, Fig. 2.6 (b), in which $\boldsymbol{x}^d(t)$ must actively search to free the parts and start over.

When assembling parts with our hands, we overcome both challenges easily. Methods tried to reproduce these skills such as Newmann's horizontal spiral search in the hybrid force/position paradigm [47]. Today, literature on assembly is extremely rich and diverse, and branched into a spectrum of approaches that continue to improve the usability and generality of the solutions. The example from Fig. 2.6(a) is a good mental baseline when reviewing these works. Within each of the following sections, we first present early approaches that introduced important concepts and terminology and move on to the most recent works.

2.2.2. Skills, Primitives and Analytic Approaches

This section reviews methods for deriving assembly skills using mostly hand-crafted solutions. For simple geometry, literature provides some analysis for the problem of force and form closure of the previous section. We then make a transition to primitives and skills that target the re-use of once programmed solutions.

Compliance Matrices and Contact States

Shimmels and Peshkin derived analytical solutions for manipulator admittance that could correct part misalignments with error-reducing motion [46]. They called part combinations on which this was possible *force-assemblable*, and derived conditions for success. Later work extended this theoretic analysis to friction [48] and proofed the derivation of suitable admittance controllers for multi-contact constellations for the planar case [49] and spatial case [50]. Investigating the contact state space between parts, the work of Xiao et al provided a symbolic description of part tilting and misalignments [51], [52]. They proposed algorithms to compute *contact formation graphs* given the part geometries [52]. A more recent enhancement is tractable for 3d objects [53]. Meeussen et al [54] use this contact formation graph in their *compliant task generator*. Automatically deriving hybrid force/position controllers reduced the programming to user-specified twist and wrench commands.

Stemmer et al presented an analytical method that relies on spring-damper parameterized compliance of the robot to obtain insertion trajectories for planar parts [55]. Alignment strategies exploit part contours, which are given as continuous, differentiable functions. An object localization handles initial part uncertainty and provides starting points within *regions of attraction* through shape matching [56]. Recent work indicates that following target motion and setting stiffness to zero in perpendicular axes can be sufficient for simple insertions [57], such that contact forces guide assemblies.

Skills and Primitives

Strip began composing skills out of *task-level primitives* in the late 80s [58]. His primitives based on the analysis of contact forces, friction cones, and the clever selection of target points for force application. They already generalized to object classes, such as rectangular, triangular, and elliptical pegs.

Onda et al [59] used a robotic manipulator to steer objects in a rudimentary simulation and extracted force/motion strategies while observing how operators switched between contact states. Their later work [60] combined this approach with a skill library [61] that chains primitives such as 'move-to-touch', 'rotate-to-level' and 'rotate-to-insert' for a hybrid position-force control in Cartesian space. Embracing a wider concept, Morrow and Khosla proposed a *sensorimotor layer* to combine sensing and action [62]. Within this layer, a library of *sensorimotor primitives* should cover up to twenty possible relative part motions which should serve as domain-general building blocks in composing task-specific skills [63]. Later work expanded to complete assembly sequences [64], [65], [66].

More recent work in this field continues this concept: Wahrburg et al formulate templates for skills with compliant trajectory execution in state machines [67]. Similar to Morrow and Khosla's early proposition, the skills are located between high-level task and low-level control and users parametrize templates explicitly in pose-wrench space.

Encapsulating some of this complexity, Thomas et al introduced *LightRocks*, a UML/P-based language for robot programming [68]. In this framework, skills are coded by domain experts and composed by shop floor workers in a state chart-like model [69]. An even higher grade of automation was introduced recently for generating and executing assembly plans from CAD files [70], albeit limiting the part mating commands for peg-in-hole types to Newman's spiral search.

2.2.3. Constraint-Based Programming

In his early *control strategy synthesis* [8], Mason presented formal models for describing compliant motion primitives in form of actuator signals to high-level sensor input with the help of two pairs of force-velocity constraints: The control strategy synthesis shall find *artificial constraints* of the end-effector, such that a goal trajectory is reproduced consistently given the *natural constraints* of the task. Lozano-Perez et al [71] showed applications for fine-motion strategies for peg-in-hole assembly. De Schutter et al [72] added the concept of tracking directions and further developed termination conditions.

Task Frame Formalism (TFF)

Basing on these works, Bruyninckx and De Schutter formulated what is known as the *Task Frame Formalism* (TFF) [73]. Strongly embedded into the hybrid force/position control paradigm from Section 2.1.2, they complete Mason's approach with a formal definition of task frames (TF) and contribute to the interface layer between task planning and force/position control. Their concept builds on the principle of vanishing virtual work:

$$\boldsymbol{\omega}^T \boldsymbol{m} + \boldsymbol{v}^T \boldsymbol{f} = \mathbf{0}\,, \tag{2.13}$$

in which the twist of the manipulated object's instantaneous motion with linear and angular velocity $[\boldsymbol{v}\ \boldsymbol{\omega}]^T$ generates no power against the wrench of forces and moments $[\boldsymbol{f}\ \boldsymbol{m}]^T$ for the contact situation [73]. In the TF formalism, Eq. (2.13) holds for all contact situations between rigid bodies, independent of reference frames and part complexity [73]. Reaction forces $\boldsymbol{f}$ are, however, assumed to be frictionless. Three requirements are essential in modeling with the TF formalism:

(i) Geometric compatibility: The goal is to find a suitable TF that models the motion constraints with up to six orthogonal axes.
(ii) Causal compatibility: Programmers specify a task with velocity setpoints for the velocity-controlled TF directions and force-setpoints for the force-controlled TF directions. This is equivalent to Mason's artificial constraints.
(iii) Time-invariance: The TF moves and adapts with time to maintain geometric and causal compatibility. Actions stop upon meeting termination conditions.

Practical use of the TF formalism, however, often requires the relaxations of these requirements. A trade-off between modeling effort and control effort has to be made, such that slightly incompatible TFs can be beneficial if the control can cope with these errors. Additionally, the setpoints can also be specified for future contacts instead of the current ones.

Kröger et al [74] provide general transformation chains for the most general TFF cases: in the robot hand, with respect to a fixed frame, and with respect to an external frame.

Their investigations allow to specify several sensor sources in different coordinate systems and provide the necessary transformations to collect all quantities in the TF for control. During execution, the TF is moving with the rigid body of interest and maintains the geometric compatibility so that the causal compatibility does not change and setpoints for velocity and force control stay constant. Limitations of the TF formalism mainly arise when no six axial and polar axes can be found to maintain geometric and causal compatibility during execution [73], which is a direct consequence of binding the task to a single TF.

iTaSC and eTaSL

Instantaneous Task Specification and Control (iTaSC) [75], [76] replaces the TFF with multiple, pair-wise *feature frames* that allow modeling the task as a composition of individual constraints. To this end, feature frames are attached to objects and allow for a more detailed focus on where a possible constraint takes action, e.g. surfaces, edges, vertices. A general constraint involves two feature frames on two distinct objects. Several of these partial descriptions then form the overall task model. This modeling procedure leads to a constrained optimization problem, in which a numerical solver minimizes a user-specified objective function subject to a set of constraints [77]. A *loop closure equation* takes geometrical uncertainty into account and inequality constraints allow the inclusion of joint limits for the control output. The solution is a velocity-resolved control reference for the robot's actuators. The optimization problem permits two distinct time horizons:

(i) step-wise, embedded into a reactive control scheme
(ii) or globally, embedded into offline trajectory planning for the full task.

Decre et al [77] add user-specific time horizons through trajectory constraints to offer trade-offs between computational speed and optimality of the two. Halt et al [78] and Nägele et al [79] build on the iTaSC framework to compose generalized, elementary sub skills for assembly and show the snap-mounting of electronic components and the insertion of a car door handle.

Further enhancing the concept of iTaSC, Aertbeliën et al [80] propose the more recent expression graph-based Task Specification Language (eTaSL). Their corresponding task controller implementation (eTC) wraps much of iTaSC's complexity and targets practical use within common robotics frameworks [17]. Pane et al [81] apply this method to a rotor assembly, in which users specify constraint types in Computer-Aided Design (CAD). A reasoning step then infers sensor-based robot skills, using an ontology to translate eTaSL compatible constraints into the optimal control framework of eTC. Vergara et al [82] bridge the gap to probabilistic imitation learning (cf. Section 2.2.5) by using Probabilistic Principal Component Analysis (PPCA) [83] to capture motion demonstrations from kinesthetic teachings in a statistical model. The connection to constraint specifications of eTaSL leads to partially transferable skills to new target positions, in which imitation learning covers wider workspace motions and constraint-based optimal control targets the last centimeters.

2.2.4. Motion Planning

Considering part assembly as a geometric planning problem implies finding collision-free paths from initial to goal configurations. This, however, becomes challenging for

low clearances. Instead of sampling collision-free states, *Kinodynamic motion planning* [84] samples admissible controls and uses state propagation under dynamic constraints by numerically solving a differential equation. For each sample, this equation has the form

$$\dot{\boldsymbol{s}} = f(\boldsymbol{s}, \boldsymbol{u}) \tag{2.14}$$

in state-space representation with state space $\boldsymbol{s}$ and control vector $\boldsymbol{u}$. If f is difficult to obtain analytically, a physics simulator models and solves Eq. (2.14) numerically [85]. If the new state fulfills user-specified costs, then the sampled control becomes the new end-point to the randomly growing roadmap. The planner terminates once $\boldsymbol{s}$ lies within a pre-defined end-game region. Kinodynamic motion planners include e.g. the extensions to Rapidly Exploring Random Trees (RRT) [85] and Probabilistic Roadmaps (PRM) [86].

Whether the planned trajectory of controls is successful on the real setup strongly depends on the simulation's accuracy in modeling important process parameters, such as friction. Since the trajectories are executed in an open-loop fashion, more robust approaches incorporate uncertainty during planning [87].

Belief Space Planning

Wirnshofer et al propose Belief Expansive Space Tree (B-EST) [88] as a belief space version of Kindel et al's kinodynamic planner [89]. They add Gaussian noise to the grasp transform of the active assembly part and model the belief of poses and twists as a joint distribution. Approximating this belief state with many particles, they sample random controls in form of piece-wise linear velocities $\dot{\boldsymbol{x}}^d$. Each particle is modeled via an impedance according to Eq. (2.12) of our introductory example, for which they use the Bullet[6] physics engine. The state propagation for each particle itself is deterministic and is executed in parallel in the simulator. If each simulated contact wrench does not exceed pre-defined thresholds, then the particle's new belief is valid and the mean of all particles forms a new node in the growing roadmap. The goal is to compute a trajectory of controls $\dot{\boldsymbol{x}}^d(t)$ such that a high portion of the distributed pose beliefs ends close to the assembly's target pose. Experiments show a superior success rate in comparison to pure kinodynamic planning on a puzzle-like assembly, a peg-in-hole problem, and a rail-mounted fuse. Their recent work [90] includes uncertainty given through elastic parts.

Constrained Motion Planning

Insertion assembly typically reduces the active part's motion and degrees of freedom along the process. Formulating motion planning explicitly under such constraints leads to the alternative of *constrained motion planning* [91]. Rodriguez et al provide an importance-based sampling technique for probabilistic planners that consider geometric constraints and efficiently search sub manifolds of the task space [92]. Users specify these constraints for every two parts with a relational positioning methodology [93].

Jäkel [94] combines constrained motion planning with the intuitiveness of the Programming by Demonstration (PbD) paradigm [95] and provides a hybrid between both approaches. Through observing human teachers with data gloves and magnetic object trackers, he automatically deduces constraint sets from demonstration and constructs

[6] https://github.com/bulletphysics/bullet3, accessed 7.8.2020

task-specific strategy graphs. He then uses efficient search heuristics and constraint relaxation to plan with spatial, force, and temporal constraints in these graphs to generate motion plans for robots of different morphology. Using human skill and experience is a powerful source for learning models from demonstration. The following Section 2.2.5 and Section 2.2.6 review important methods under this paradigm that can be used for assembly.

2.2.5. Probabilistic Skill Learning from Demonstration

Learning from human demonstration is a strong alternative to explicit task modeling and marks an upwards trend in robotics over the last years [96]. The term Learning from Demonstration (LfD) in general can be considered a superclass of methods in robotics with the core idea to learn from the performance of human teachers [96]. Common synonyms to refer to this approach are Programming by Demonstration (PbD) [5], [95], which coined the field, Imitation Learning (IL) [97] with a background in physiological neurosciences, and Behavioral Cloning (BC) [98] from the wider field of machine learning.

A common incentive among variants is to learn a policy from human-generated data. This thesis' focus of insertion assembly requires policies that produce low-level, subsymbolic trajectories that robot controllers can execute through force or motion setpoint tracking. For the variant described in this section, probabilistic models shall learn despite the variance in human demonstrations.

Gaussian Mixture Models (GMM)

Gaussian Mixture Models (GMM) are a suitable means of representing human recorded datasets. The recordings typically constitute measurements that rather belong to the input space $\boldsymbol{x}$ or the output space $\boldsymbol{y}$, respectively. Both form vectors in a multivariate case. We use the notation of [99] in the setting of the multi-variate case described in [100]. From a probabilistic viewpoint, these signals constitute random variables, and the idea is to represent the dataset as a joint probability density

$$p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{k} \alpha_i p_i(\boldsymbol{x}, \boldsymbol{y}), \tag{2.15}$$

with $\sum_i \alpha_i = 1$ as the priors of a mixture of k Gaussians. The individual, c-dimensional distributions have the form

$$p_i(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{\sqrt{(2\pi)^c |\boldsymbol{\Sigma}_i|}} \exp\left[-\frac{1}{2}(\boldsymbol{v} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{v} - \boldsymbol{\mu}_i)\right], \tag{2.16}$$

with the joint vector of variables $\boldsymbol{v} = [\boldsymbol{x}^T \boldsymbol{y}^T]^T \in \mathbb{R}^c$ and the corresponding means and covariances for each kernel

$$\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_{i,x} \\ \boldsymbol{\mu}_{i,y} \end{bmatrix} \in \mathbb{R}^c \quad \text{and} \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_{i,xx} & \boldsymbol{\Sigma}_{i,xy} \\ \boldsymbol{\Sigma}_{i,yx} & \boldsymbol{\Sigma}_{i,yy} \end{bmatrix} \in \mathbb{R}^{c \times c}. \tag{2.17}$$

Both can be learned iteratively from the dataset through applying the Expectation Maximization (EM) algorithm [101], [102], which yields a locally optimized fit of the Gaussian

kernels to the dataset. The number of kernels needs to be specified beforehand and does require a certain knowledge of the task.

For regression, using the conditional probability density $p(\boldsymbol{y} \mid \boldsymbol{x})$ allows us to compute an output $\boldsymbol{y}$ for a given input $\boldsymbol{x}$. Through learning a joint probability, GMMs make no distinction between input and output, and the strength of the approach is its flexibility to use arbitrary input-output constellations through conditioning on individual dimensions of $\boldsymbol{x}, \boldsymbol{y}$ [99]. A frequent application is to learn a joint distribution by including time explicitly as a random variable t and then using $p(\boldsymbol{y} \mid t)$ to describe trajectories over time [103].

Gaussian Mixture Regression (GMR)

Gaussian Mixture Regression (GMR) models the conditional probability density $p(\boldsymbol{y} \mid \boldsymbol{x}) \sim \mathcal{N}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{\Sigma}}_{yy})$ to obtain a Least Squares Estimate (LSE) for a given input [102]. Using the previously fitted means and covariances from Eq. (2.17), the estimate and its variance become (see e.g. [100], [103]):

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{k} h_i(\boldsymbol{x}) \left(\boldsymbol{\mu}_{i,y} + \boldsymbol{\Sigma}_{i,yx} \boldsymbol{\Sigma}_{i,x}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_{i,x}) \right) \tag{2.18}$$

and

$$\hat{\boldsymbol{\Sigma}}_{yy} = \sum_{i=1}^{k} h_i^2(\boldsymbol{x}) \left(\boldsymbol{\Sigma}_{i,yy} - \boldsymbol{\Sigma}_{i,yx} (\boldsymbol{\Sigma}_{i,xx})^{-1} \boldsymbol{\Sigma}_{i,xy} \right) \tag{2.19}$$

with

$$h_i(\boldsymbol{x}) = \frac{\alpha_i \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{i,x}, \boldsymbol{\Sigma}_{i,x})}{\sum_{j=1}^{k} \alpha_j \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{j,x}, \boldsymbol{\Sigma}_{j,x})}. \tag{2.20}$$

h_i denotes the normalized probability of $\boldsymbol{x}$ belonging to each Gaussian [102], such that $h_i \in [0, 1]$, $\sum_i h_i = 1$. GMR is often used for trajectory learning in motion and force-based manipulation [100], [104], [105], [106]. Tang et al [107] use this approach to describe an admittance controller as a learned probabilistic mapping from sensed contact wrenches to twist commands. They record the demonstrations of a peg-in-hole experiment with a specially designed teach device. Note that Eq. (2.18) yields deterministic estimates for robot control. Kronander et al [108] use sampling from the conditional probability distributions $\mathcal{N}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{\Sigma}}_{yy})$ to introduce randomness in comparison to LSE and thus overcome deadlocks during assembly.

Hidden Markov Models (HMM)

As an alternative to including time explicitly within the joint probability density of GMM, Hidden Markov Models (HMM) can be used to feed sequential information into the regression function of GMR. HMMs are temporal probabilistic models and resemble GMMs in that they model the dataset with joint probability densities. But in addition, they learn sequential information in the form of state transitions that can be harvested to make temporal predictions.

Using the notation of [104], the HMM is described by $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, in which $\boldsymbol{\pi} = \{\pi_i\}$ denotes the initial state probability vector with $1 \leq i \leq k$, and $\boldsymbol{A} = a_{ij}$ is a matrix describing the transition probability from state i to state j with $1 \leq i, j \leq k$. The

mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ represent a set of c-dimensional Gaussians $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, one for each state i, analog to Eq. (2.17). In the context of HMMs, the recorded dataset constitutes a set of observation sequences $\boldsymbol{O} = \boldsymbol{O}_1, \boldsymbol{O}_2, \ldots, \boldsymbol{O}_T$, in which $\boldsymbol{O}_t$ is an input-output pair $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ at time t. The goal is to adjust the model parameters $\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ to optimize $p(\boldsymbol{O} \mid \lambda)$, i.e. making the HMM best explain the recorded sequences. This step can be achieved iteratively with the Baum-Welch algorithm or the EM algorithm, see e.g. [109], and the result is a local best fit of the model to the dataset. Al-Yacoub et al [110] use this intermediate result to identify generic state transitions during a peg-in-hole assembly task and cluster force-torque signals to get insights into human skill characteristics.

Having the HMM's parameters, Calinon et al [104] and Rozo et al [105] use the *forward variable* [109] to include temporal context into the regression function of Eq. (2.18) in the form of

$$h_{i,t}(\boldsymbol{x}) = \left(\sum_{j=1}^{k} h_{j,t-1} a_{ji} \right) \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{i,x}, \boldsymbol{\Sigma}_{i,x}). \tag{2.21}$$

While $h_i(\boldsymbol{x})$ from Eq. (2.20) described the probability of $\boldsymbol{x}$ belonging to each Gaussian, $h_{i,t}(\boldsymbol{x})$ from Eq. (2.21) describes the probability of being in state i at time t with including how likely that is given the previous observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t$ up to the present.

GMR with this approach encapsulates both spatial and sequential information probabilistically. Stability in these HMM-GMR combinations outside the taught regions can be achieved with additional attractor methods [104]. The next section reviews established alternatives using dynamical systems for motion representation with primitives.

Dong et al [111] use a slightly different approach in which they combine an HMM with Locally Weighted Regression (LWR). In contrast to GMR, LWR is a non-parametric, memory-based method and requires training data during control applications. LWR performs a regression around a query point of interest by using kernel functions to weigh the distance to local training data. Their HMM learns discrete contact states of peg-in-hole assembly from demonstrations in a virtual, haptic simulation environment. Upon detecting such states during task reproduction on a 3-degree-of-freedom mechanism, they use LWR on a subset of the training data as identified by the HMM for controlling angle corrections to measured forces and torques.

2.2.6. Imitation Learning with Primitives

Learning skills for regions outside of the human demonstrations requires sufficiently time and space-independent, parametric models. With the incentive to capture and generalize motion characteristics the design of dynamical systems for movement imitation [112] inaugurated an important field for skill learning.

Dynamic Movement Primitives (DMP)

Dynamic Movement Primitives (DMP) [113] are a common method of using dynamical systems for motion representation of coordinated multi-degree of freedom mechanisms. They were first introduced by Ijspeert et al [112] and termed DMPs in the work of Schaal [114], and have been enhanced in further works [115], [116]. To avoid dependency on robot morphology, a common way is to formulate DMPs in task space and use end-effector control for the inverse kinematics problem, see e.g. [116]. The generated

trajectories keep their motion characteristics for different goal positions. We base or following short overview on the notation of [116]. The basic idea is to describe motion of interest $\boldsymbol{x}(t)$ of a state vector $\boldsymbol{x}$ as a set of ordinary differential equations, in which each dimension x is defined by an individual spring-damper system

$$\tau^2\ddot{x}(t) = K(g - x(t)) - \tau D\dot{x}(t) + (g - x_0)f(s(t)) \quad . \tag{2.22}$$

Stiffness K and damping D are hand-tuned constants, whereas the non-linear perturbation f is to be learned from observation. Eq. (2.22)) is referred to as *transformation system*. One of the decisive features of DMPs is the generalization of the learned motion characteristics for each dimension when using new targets g and the start states x_0. The duration of the playback is adjusted by choosing a specific time scale τ. The motion characteristics, which is comprised in f depends on a dimensionless phase variable s for the course of the skill. Common for all transformation systems, s is described by the following *canonical system*

$$\tau\dot{s}(t) = -\alpha s(t) \quad , \tag{2.23}$$

with the scaling factor τ and a constant α, relating phase and real-time with an exponential decay from 1 to 0. This assures a coordinated, timeless parameterization of the individual transformation systems, each represented by an instance of Eq. (2.22). Each non-linear perturbation is modeled by

$$f(s) = \frac{\sum_i \omega_i \psi_i(s) s}{\sum_i \psi_i(s)} \quad , \tag{2.24}$$

with $\psi_i(s) = \exp(-h_i(s - c_i)^2)$ representing an a-priori chosen number of Gaussian basis functions with constant center c_i and constant width h_i. Using the DMPs to obtain motion-based skills boils down to learning the adjustable weights ω_i. For this purpose, a skill-defining trajectory $x(t), \dot{x}(t), \ddot{x}(t)$ is recorded from human demonstration for a certain time span. Reformulating Eq. (2.22) leads to a straight-forward computation of discrete samples

$$f^d(s(t)) = \frac{-K(g - x(t)) + \tau D\dot{x}(t) + \tau^2\ddot{x}(t)}{g - x_0} \quad . \tag{2.25}$$

This step can be interpreted as inverse dynamics computation, in which the desired disturbance f^d is fully defined by the observation $x(t)$ over time. The weights of f are then learned in a supervised manner, such that $f(s)$ best agrees with the data samples $f^d(s)$ observed. Minimizing $\sum_s (f^d(s) - f(s))^2$ can be solved e.g. with multivariate linear regression.

When deploying the DMP, first Eq. (2.23) and then each dimensions' Eq. (2.22) are numerically integrated with a user-specified τ. Obtaining $x(t)$ corresponds to finding numeric solutions to initial value problems with the initial conditions $x(0) = x_0, \dot{x}(0) = \dot{x}_0$. A simple implementation could transfer Eq. (2.22) into a set of first-order ODEs and obtain the state vector $[x(t), \dot{x}(t)]^T$ incrementally with the forward Euler method. The obtained $x(t)$ and $\dot{x}(t)$ are then passed as reference signals to the controller. When chaining, the goal states of a DMP initializes the start states of the succeeding DMP.

Combining the DMP approach with probabilistic components allows for capturing of variance in human demonstrations. Ding et al [117] first use GMR from Section 2.2.5 to learn concise trajectories from human demonstration and then use DMPs to obtain a parametric model for generalization to different start and end positions across the workspace.

For insertion tasks with more complicated contact dynamics, however, motion alone is usually not sufficient to model characteristics in human demonstrations. The integration of force profiles plays a decisive role in increasing robustness.

Learning Force Profiles in DMPs

Pastor et al [118] use an online adaptation of trajectories through an additive term in the transformation system of Eq. (2.22) that computes an error-proportional disturbance for target force-torque profiles. They obtain these profiles as recorded sensor traces from previous executions with the idea of building a growing associative memory for similar tasks [119]. Another approach is to generalize force-torque profiles in time via the phase variable s. In contrast to a direct replay of recordings and setpoint tracking, the idea is to capture the characteristics of the sensor signals analog to the capturing of motion profiles into a parametric model. Spatial generalization on the other hand is usually achieved through formulating the DMPs in task space coordinates. This leads to a formulation similar to Eq. (2.24) for modeling forces $f(s)$ and torques $t(s)$ as linear combinations of radial basis functions:

$$f(s) = \frac{\sum_i \omega_i^f \psi_i(s)}{\sum_i \psi_i}, \qquad t(s) = \frac{\sum_i \omega_i^t \psi_i(s)}{\sum_i \psi_i}. \tag{2.26}$$

The difference is that they are not embedded into the spring-damper system of Eq. (2.22) as a perturbation, but instead are directly used as reference profiles in end-effector force controllers and are tracked in parallel to the motion trajectories, e.g. with a proportional-integral (PI) controller [120]. For velocity-actuated robots, the force-controlled component is usually mapped to motion space with a compliance matrix. Three dimensions of each $f(s)$ and $t(s)$ are required to capture six-dimensional profiles.

Nemec et al [121] and succeeding works [120], [122], [123], [124] use this mechanism for peg-in-hole assembly tasks. Kramberger et al [123] use LWR from Section 2.2.5 as a memory-based learning method around query points of several demonstrations to increase performance.

Abu-Dakka et al [120] and Savarimuthu et al [124] apply a quaternion-based DMP formulation [125] and use the phase modulation mechanism to slow down the execution for better adaptation to the target force profiles whenever execution speed is too fast for setpoint tracking. They additionally apply Iterative Learning Control (ILC) [126] to closer match the force profiles over repetitive executions. Using teleoperation avoids the task replay of [123] to obtain unbiased force profiles in the first run.

2.2.7. Reinforcement Learning

Our revision so far ordered the approaches roughly to engineering effort in task modeling. In this last section, works from the field of Reinforcement Learning (RL) [127] share the same goal of obtaining policies for assembly but do so in a rather different way. Especially in game-like environments, agents trained with RL can reach a superhuman level through extensive self-play [128], [129]. RL differs substantially from methods from previous sections in that agents find suitable policies themselves, albeit requiring a hand-engineered reward function to learn through trial and error. Inverse RL in the context of apprenticeship learning can obtain this reward function from human expert

demonstration [130], [131], closing the circle of imitation learning. Reinforcement learning for industrial assembly is not new. Gullapali et al have shown peg-in-hole insertion in the early 90s, using neural networks of two hidden layers for admittance control-based strategies [132], [133]. Modern approaches mainly rely on very deep neural networks that are more powerful, complex function approximators. We briefly highlight how selected, recent works apply RL for similar or relevant tasks.

Guided Policy Search (GPS)

A classical challenge in RL is the sample inefficiency for high-dimensional state spaces. Especially for articulated robots, this results in high system interaction time. Levine at al [134] apply Guided Policy Search (GPS) [135] as a sample-efficient method to these settings. Although not explicitly targeting assembly, their approach is suitable for contact-rich tasks. They try to learn the parameters of a policy

$$\pi(\boldsymbol{u}_t \mid \boldsymbol{x}_t) \tag{2.27}$$

as a probabilistic mapping from robot state $\boldsymbol{x}_t$ (joint positions, velocities, target objects) to low-level actions $\boldsymbol{u}_t$ (joint torques). Instead of learning the policy directly with an RL method, their approach combines two components: The first component is the training of time-varying, linear Gaussian controllers for trajectory optimization under unknown dynamics. Basing on iterative Linear Quadratic Regulators (iLQR) [136], which is a common method in optimal control for trajectory optimization, they additionally need to estimate the dynamics iteratively, solving linear-quadratic Gaussian (LQG) problems and using samples from the previous rollout in each step. A cost based on the Kullback-Leibler (KL) divergence between succeeding trajectory distributions prevents divergence into infeasible state spaces. The second component uses GPS to find a non-linear policy as a combination of several of these controllers. Since Gaussian controllers alone are not robust to changes of start or target positions, the idea is to obtain a more generalized representation through parameterizing the deep neural network π from Eq. (2.27) with supervised learning based on samples from these controllers. Generalization implies a difference in state distributions in the Gaussian controllers and the final policy. Their approach thus uses Dual Gradient Descent (DGD) to solve both trajectory re-optimization of the Gaussian controllers and policy parameter learning jointly in an alternating fashion to converge to suitable state distributions.

On the tasks presented, this works in roughly 20 to 40 rollouts and can handle the non-linearities of interaction dynamics. Note, however, that their method needs the manual specification of *intermediate waypoints* for tasks where a non-straight motion to the target is required.

GPS and Motion Planning

Similar to this work, Thomas et al [137] use the iLGQ-based policy search algorithm from [135] but combine it with motion planning to achieve greater workspace independence of their policies. As described in Section 2.2.3, geometric motion planning computes a continuous path from an initial configuration to a goal configuration in the robot's free configuration space, using CAD data of assembly parts as obstacles. Their approach models the cost function in such a way that it motivates to track an initial motion plan as

a reference trajectory. RL shall learn contact dynamics that are not anticipated by the motion plan. In the setting of iLQG, the first pass is initialized with a proportional derivative (PD) controller that tracks this trajectory. During execution, their neural network policy uses an attention mechanism that selects time windows of normalized trajectory inputs to compute the directions of the next actions.

Levine et al [134] had the inherent drawback of learning within the highly non-linear configuration space of the robot, generating the need to train multiple controllers to sufficiently cover the task's workspace. By including a freshly generated motion plan into the policy network for each task execution, Thomas et al [137] could remedy this drawback. An alternative is using RL in part-centered task space directly, whose dimensions are strongly reduced in comparison to the full robot's state space that is governed by non-linearities of kinematics and dynamics. Actuation is then delegated to Cartesian robot controllers for setpoint tracking.

Deep Q-Learning

Inoue et al [138] use deep Q-learning in such a setting for peg-in-hole with micrometer clearance. In contrast to policy gradients, Q-learning is often embedded into a Markov decision process and finds optimality estimates (Q-values) for discretely defined actions, leading to a policy

$$\pi(\boldsymbol{s}) = \text{argmax}_{\boldsymbol{a}} Q(\boldsymbol{s}, \boldsymbol{a}) \tag{2.28}$$

that chooses the best action $\boldsymbol{a}$ at each state $\boldsymbol{s}$. They model π as a deep recurrent neural network to include sequential state observation against control delays and possible signal noise. The state space is composed of measured force-moment signals and a horizontal position in a grid. The elementary action space is composed of target forces and rotations along unit axes, which they apply as setpoints in a hybrid position/force controller. For each learning episode, an action thread creates a replay buffer with state-action pairs and rewards during insertion trials, and a learning thread updates the network parameters, randomly sampling from the replay buffer. Distance and depth are chosen as penalties during the search phase and insertion phase, and a positive reward motivates a completion within a minimal number of steps for successful episodes. They apply an ϵ-greedy policy for the action thread to balance exploration and exploitation, in which the best action (with the highest Q-value) is taken with probability $1 - \epsilon$ and the rest is randomly chosen with probability ϵ from the complete action set. After precisely calibrating the setup and determining suitable magnitudes of the force actions, their approach shows successful insertion with only $20\,\mu\text{m}$ and $10\,\mu\text{m}$ part clearance after 200 training episodes on the real setup with up to $3\,\text{mm}$ translational and $1.6°$ rotatory offset.

Visual Reward Functions

Based in a similar setting of Q-learning, Schoettler et al [139] use a combination of a fixed hand-specified controller $\pi_H(\boldsymbol{s})$ and a neural network policy $\pi_\theta(\boldsymbol{s})$ for electronic plug insertion. The idea is to use RL on top of a conventional controller to increase sample efficiency and reduce system interaction time. Their approach chooses actions with

$$\boldsymbol{a} = \pi_H(\boldsymbol{s}) + \pi_\theta(\boldsymbol{s}) \tag{2.29}$$

at $10\,\text{Hz}$, which control the robot's end-effector pose through inverse kinematics with joint-space impedance. π_H is obtained through demonstrations with a gamepad. The

neural network parameters are learned with the stochastically exploring Soft Actor Critic (SAC) algorithm [140]. Considering several combinations, the best working reward function uses a 32×32 greyscale image of the task with pixel-wise $L1$ distance to a goal image, weighing-in divergence from the prior demonstrations. Their results show successful insertion with an uncertainty of up to $1\,\mathrm{mm}$ to the ground truth. For simple insertions with one primary axis, this combination of visual input and prior demonstration can circumvent RL's typical laborious reward shaping. The advance of image processing pipelines with regional Convolutional Neural Networks (CNN) [141] allows a semantic image annotation for tasks that provide sufficient visual characteristics. Ding et al [142] use this approach to improve their force-based search with hole detection and pose estimation from images of an RGB-D camera.

2.2.8. Discussion

The last sections showed a broad and diverse field that spans several decades of research. Individual philosophies branched out and we classified six important directions with methods close to robotic assembly.

Analytic approaches and hand-designed skills target the reuse of primitives to compose robot programs. They are characterized by a precise knowledge of task geometry and mostly require an engineer's background in their parameterization. Using task constraints and formalisms led to constraint optimization solvers that can tackle more complicated, albeit explicitly specified tasks. Putting numerical simulation into the focus, motion planning circumvents some complexity in task set-up but requires realistic parameters to simulate process dynamics. Characteristic for these approaches is the open-loop execution once a plan is computed. An alternative is planning in constraint submanifolds, whose constraints can be derived from human observation. This changes the philosophy to programming by demonstration and enables non-professionals to teach tasks and provide datasets for machine learning. Probabilistic approaches generalize better with more demonstrations. Dynamic movement primitives achieve generalization through attractor dynamics and learn time and space-invariant trajectories. Forces profiles are handled as time-coupled overlays and fewer demonstrations are required in comparison to probabilistic imitation learning. Reinforcement learning contrasts with these approaches in that algorithms heavily rely on exploring by trial and error. Engineers have less control over convergence and reward shaping is necessary in most cases.

The following analysis sets important goals and evaluates the strengths and drawbacks of the individual directions.

2.3. Analysis

Previous sections reviewed related works and presented various techniques and directions for robotic assembly. However, not all of them are directly applicable to our settings and problem statement and require a high-level reflection concerning the goals of this thesis. Section 2.3.1 analyses the research directions under specific criteria to highlight their strengths and drawbacks. Section 2.3.2 draws a conclusion based on this analysis and sketches this thesis' approach.

2.3.1. Goals and Limitations

Three goals are of special importance: The *Applicability* to robots and control approaches from Section 2.1 and the example from Section 2.2.1; the suitability for *Offline Programming* within industrial settings; and low *Engineering Effort* for task programming as a quality measure for ease of use. We evaluate the research directions from Section 2.2 under the criteria that we formulate for these goals. This evaluation shall offer a general overview with the following grading:

$+$ Generally a strength of this direction and mostly applicable without difficulty
0 Mostly indifferent without positive nor negative tendency
$-$ Rather a weakness of this direction and causing some difficulty for application

Table 2.1 to Table 2.3 summarize the results.

Applicability

(1) Compatibility with motion-actuated systems Section 2.1.1 posed *motion-actuated* systems as a requirement for assembly methods. These robots are non-backdrivable in their joints but become compliant at their end-effectors through imitating active compliance by control. Suitable control schemes build upon specifying motion-force setpoints either explicitly in Hybrid Force/Position Control Section 2.1.2 or implicitly in Admittance Control Section 2.1.3. Analytical approaches with composing skills out of primitives and constraint-based programming integrate well into these schemes: Parameterizing setpoints by hand or optimizing them in constraint solvers aligns with the thinking in force and motion subspaces. Constraint motion planning can search the motion-actuated joint space of the robot and thus also fits well. Additionally, kinodynamic motion planning can compute the trajectories of controls in the admittance scheme. The learning from recorded datasets in DMPs and probabilistic LfD is itself robot-independent. For active compliance, kinesthetic teaching applies at the robot end-effector. If that is less suitable, teleoperation can be used as an alternative [143]. For Reinforcement Learning, active compliance requires measurements to assure controller stability and safety during the interaction. This slows down task learning and increases system interaction time.

(2) Suitability for low clearances and tight fits Both are characteristic of the assembly tasks we consider. Analytically derived skills and primitives depend on a good estimation of friction, and favor the separation of motion and force control. In tight fits, the separation of primary directions for motion and force becomes fuzzy. Human dexterous manipulation contains much trial and error, leading to a non-trivial parameterization if converted into hybrid force/motion templates. Constraint solvers also build on a mathematical formulation of friction, which must be sufficiently adequate to cover real-world execution in highly constraint tasks. Since kinodynamic motion planning relies on simulation, the physics environment must support the modeling of such tight fits, which closely correlates with numerical contact instability. In contrast, probabilistic imitation learning appears advantageous, but the models must capture the recorded datasets adequately. GMR e.g. will lose trial-and-error strategies through learning the average. Dynamic movement primitives require the learning of force profiles for these tasks, because motion is barely visible and most of the human intention is measured through contact forces. In comparison, RL's exploration is well-suited.

(3) Suitability for arbitrary masses and dimensions Analytical methods and constrained-based programming are mostly independent of object masses and dimensions, although use cases tend to focus on handy objects. If masses and moments of inertia become substantial against friction, kinodynamic motion planning requires these as additional values for simulating state propagation. Heavy and bulky components are in general a limitation for learning from demonstration-based techniques. If the robot control offers load compensation [144], these techniques might still be applicable in some cases. Heavy loads quickly build up high reaction forces in contact transitions, such that RL is further limited to very low execution speeds for exploration in these cases.

Table 2.1.: Applicability

	(1)	**(2)**	**(3)**	Selected works
Skills, Primitives and Analytic Approaches	+	−	+	[64], [68], [70]
Constraint-Based Programming	+	−	+	[73], [80]
Motion Planning	+	−	0	[92], [94], [88]
Probabilistic Skill Learning from Demonstration	0	+	−	[108], [107]
Imitation Learning with Primitives	0	0	−	[121], [120], [122]
Reinforcement Learning	−	+	−	[134], [138], [139]

Offline Programming

(4) Suitability for simulation and transfer The requirement of offline programming means having no access to the system during task programming. For analytic approaches, this is less of a restriction, and the composing of skills out of primitives already relies on models. Constraint-based programming operates on a similar level of abstraction and CAD and simulation environments build the setting for task design. Programming by Demonstration is more restricted: Approaches use real parts or replicas with magnetic trackers for teleoperated programming and force-based DMPs mainly leverage kinesthetic teaching.

The important question, apart from whether the approaches can be carried out in simulation, is how much they depend on accurately modeling the process parameters. In robotics research, the transfer from simulation to real-world settings is classically referred to as the reality gap [145], and tight-fitting assembly operations are particularly challenging in simulation [146]. The realism of contact forces and friction in simulation directly impacts the success of open-loop approaches, such as motion planning. Sufficiently modeling the real hardware's physics or learning under uncertainty is of similar importance for RL.

(5) Handling of process uncertainty, part jamming/wedging Some sort of cognition must detect and handle deviations online to yield robust executions. Analytic approaches partially incorporate this in form of contact state transitions and through the parameterization of corrective skills. They however need engineered termination conditions. Constraint solvers allow for the inclusion of uncertainty in form of constraint relaxations, but a problem arises with open-loop control in general: If sensory feedback is not taken into consideration, then the computed controls strongly depend on the models' accuracy and

are sensitive to uncertainty. Open-loop approaches incorrectly assume the end of the task implicitly after executing their trajectories, and intermediate deadlocks are thus not anticipated. Motion planning does not include sensory feedback besides what is required for compliant control on a lower level. Planning in belief space adds tolerance to modeling uncertainty but likewise stays open loop. Probabilistic imitation learning can introduce randomness into the executions and thus overcome deadlocks through variance. With DMPs, one of the intrinsic ideas is to couple individual dimensions through a canonical system, assuring timely consistency during playback. DMPs are, therefore, bound to a specific duration that the user chooses before the execution starts, and force profiles become open-loop control signals. RL provides advantages by explicitly including sensory feedback into reactive control policies.

Table 2.2.: Offline Programming

	(4)	**(5)**	Selected works
Skills, Primitives and Analytic Approaches	+	0	[60], [68]
Constraint-based Programming	+	0	[77], [82], [81]
Motion Planning	0	−	[88]
Probabilistic Skill Learning from Demonstration	−	+	[111]
Imitation Learning with Primitives	−	−	[124]
Reinforcement Learning	−	+	[137], [139]

Engineering Effort

(6) Intuitiveness of programming This can be considered a quality measure for the different approaches. Being more intuitive means enlarging the group of potential programmers and increasing the method's applicability. The less expert thinking is required, the better. After parameterization by experts, primitives for CAD-based skill composition are a feasible method to reduce the online programmer's frequently re-occurring steps. In constrained-based programming, much complexity can be hidden behind skill frameworks so that users specify constraint types in CAD. Motion planning offers a likewise simple usage due to directly specifying goal poses for the assembly tasks. With learning from recorded performance, task programming simplifies to showing the assembly. Probabilistic imitation learning and DMPs are thus among the most intuitive ways of programming. They also solve the problem of parameterization for which hand-crafted skills and primitives require experts. The solutions obtained with RL are, in comparison, more autonomous but less intuitive. Influencing the algorithms' non-deterministic convergence requires domain knowledge for reward shaping.

(7) Independence of the task complexity Being a strong approach under this measure means having a constant engineering effort, independent of complicated workpieces and multi-directed insertion. Deriving controllers analytically from workpiece geometry e.g. does not scale well with complexity. Primitives, in contrast, stay atomic, but composing them into skills becomes more complex. Using databases and ontologies for deriving constraint types from CAD alleviates this bottleneck for approaches with constraint solvers. Similarly, task complexity hardly influences engineering effort for motion planning, and

workpiece complexity only increases computational costs for collision checking during the search. Probabilistic and dynamic imitation learning also have their strengths in this field: Programming the assembly task through showing and learning from data stays intuitive with increased part complexity. Besides hand-designed intermediate waypoints for policy search with non-direct insertions, the algorithms behind RL, in general, stay mainly constant with increasing part complexity, and no significant overhead arises.

Table 2.3.: Engineering Effort

	(6)	**(7)**	Selected works
Skills, Primitives and Analytic Approaches	0	−	[67], [69]
Constraint-based Programming	0	0	[78], [79]
Motion Planning	+	+	[94], [88]
Probabilistic Skill Learning from Demonstration	+	+	[108], [107]
Imitation Learning with Primitives	+	+	[123], [117]
Reinforcement Learning	−	+	[134]

2.3.2. Conclusion

Table 2.4.: Evaluation of assembly directions.

Criteria	Skills, Primitives and Analytic Approaches	Constraint-based Programming	Motion Planning	Probabilistic LfD	Imitation Learning with DMPs	Reinforcement Learning
(1) Compatibility with motion-actuated systems	+	+	+	0	0	−
(2) Suitability for low clearances and tight fits	−	−	−	+	0	+
(3) Suitability for arbitrary masses/dimensions	+	+	0	−	−	−
(4) Suitability for simulation and transfer	+	+	0	−	−	−
(5) Handling of uncertainty/jamming/wedging	0	0	−	+	−	+
(6) Intuitiveness of programming	0	0	+	+	+	−
(7) Independence of the task complexity	−	0	+	+	+	+

Table 2.4 shows the final overview of the considered research directions. There is currently none with strengths under all criteria and the biggest potential lies within combinations. The goal of this thesis is to combine the strengths of Skills/Primitives with probabilistic Learning from Demonstration and contribute to the field of industrial offline programming for challenging assembly tasks. The concept of skills neatly integrates with control schemes for motion-actuated systems and supports simulation and transfer. The usage of the PbD paradigm shall avoid their parameterization with expert knowledge and instead learn strategies against part jamming/wedging implicitly from demonstra-

tions in an intuitive manner. Using simulation at the core of the approach requires some form of teleoperation, and this thesis will need to tackle the reality gap for tight-fitting insertion and will thrive to circumvent the dependency on realistic and accurate process parameters.

Chapter 3 first proposes a robot-independent way of learning human-inspired skills from demonstration in simulation environments and Chapter 4 derives a new compliant control scheme to transfer these skills to motion-actuated systems.

3. Human-Inspired Assembly Skills

Following the PbD paradigm for offline programming, we combine teleoperation with simulation to provide an intuitive way of demonstrating assembly tasks. The goal is to extract strategies from these demonstrations and to design suitable models for their technical representation. This chapter presents the different steps to realize this approach.

Section 3.1 first describes our concept behind skill learning, explains the assumptions we make, and defines important terms for the remainder of this thesis. Section 3.2 then derives methods for recording human behavior in simulation environments with a focus on realizing low-clearance assembly. Data analysis of exemplary demonstrations builds the basis for modeling the skills in Section 3.3. We describe how to learn patterns in probabilistic sequences with neural networks and how to train and use them for inference. Section 3.4 combines these skills with trajectory execution and shows how to specify assembly tasks within this approach.

3.1. Concept

The mathematical representation of the strategies will be a combined vector of forces and torques $\boldsymbol{f}^h \in \mathbb{R}^6$, with which Eq. (2.12) becomes

$$\boldsymbol{f} = \boldsymbol{K}\Delta\boldsymbol{x} + \boldsymbol{D}\Delta\dot{\boldsymbol{x}} \quad + \boldsymbol{f}^h \,. \tag{3.1}$$

The superscript ^{h} shall indicate that it's the output of a human-inspired skill. This vector component takes the role of counterbalancing jamming and wedging between the parts by dynamically incorporating meaningful corrective measures. In that regard, $\boldsymbol{f}^h$ overrules the controller's restoring forces $\boldsymbol{K}\Delta\boldsymbol{x}$, whenever force and form closure occur during assembly.

Since $\boldsymbol{f}^h$ is to be learned from expert demonstrations, the question arises which sensor feedback is crucial during assembly and what, other than time, is required to ground such skills.

Klingbeil et al [147] investigated if humans visit certain contact states during assembly, which could support the hypothesis, that we build some sort of high-level contact representation to which we link specific strategies. They found that apart from some common positions, which many testers did visit, intermediate behavior was seemingly random. High variance in how the test group approached the studied problem showed that much trial and error is involved. Despite this variance, however, the executions were successful, underlining the fact that all user strategies matched concerning hidden but crucial characteristics.

Insights from physiological studies suggest the existence of forward models [148] for control and planning through the central nervous system. Miall and Wolpert [149] highlight forward models as a special type of internal model to mimic the motor system's response to outgoing motor commands. These models are assumed to enable the estimation of the sensory consequences of these motor commands in the form of state transi-

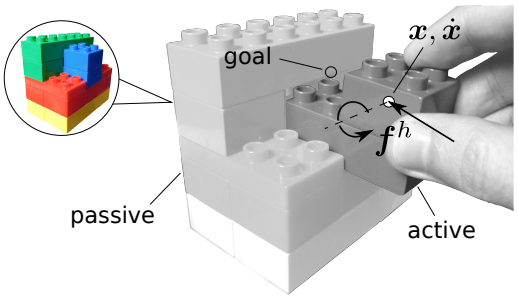


Figure 3.1.: A toy assembly will serve as an illustrative example for the concepts of this chapter. Assuming mental forward models in human control, there is a strong connection between motor control $\boldsymbol{f}^h$ and the differences of mentally expected and observed states $\boldsymbol{x}, \dot{\boldsymbol{x}}$. The idea is to learn strategies from these variables for robotic assembly.

tions. The difference between estimated states and sensed states could inform the central nervous system about external events [149].

Motivated by the idea of a sub-conscious forward model behind human strategies, this thesis focuses on $\boldsymbol{f}^h$ as the human-initiated motor command and $\boldsymbol{x}, \dot{\boldsymbol{x}}$ as the object's sensed state as crucial variables during assembly.

Consider the illustration from Fig. 3.1. The toy assembly consists of two parts, which form a compact block after joining. Although seemingly simple, the plastic parts typically have low clearance and the geometry requires dexterity for a complete insertion, making robotic execution extremely challenging. We use this assembly as an illustrative and representative example for deriving the concepts of later sections. Chapter 5 evaluates this thesis' approach to additional real-world applications. The variables $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ denote the active part's pose and spatial velocity with respect to the estimated goal pose. $\boldsymbol{f}^h$ is the combined force-torque vector of net forces, applied via the fingers. In the light of internal models, the idea is to assume that jamming, unexpected tilting, and wedging of the parts during assembly are perceived as a perturbation of the mental forward model for this task, which causes us to take counteractive strategies. The goal of this thesis is to extract and learn these intuitive strategies from the observed data stream $\boldsymbol{f}^h, \boldsymbol{x}, \dot{\boldsymbol{x}}$.

The next sections formulate this idea as a learning problem and describe two important design decisions: The usage of simulation and relying only on visual feedback during demonstrations.

3.1.1. Terms and Definitions

In the recently proposed taxonomy for distinguishing methods in Learning from Demonstration [96], and anticipating later sections, this thesis' approach could be roughly classified as
Demonstrations: Teleoperation
Learning outcome: Policy
Policy input: State
Policy output: Low-level actions
Policy class: Stochastic - time-dependent
with slight inconsistencies in the wording of policy input and output. As later sections will detail, our approach builds on methods from general machine learning for

sequence modeling, which does not necessarily fit perfectly into the state-action thinking of robotics. Industrial assembly is a core discipline of robotics and we nevertheless try the formulation in this domain.

Our approach builds upon the idea of *behavioral cloning* [98], which is frequently used synonymously with *Programming by Demonstration* [103] and the wider term *Imitation Learning* [150]. The goal is to model assembly skills in form of policies π as the solution to a supervised learning problem from expert demonstrations. Following the mathematical notation of [151], we consider trajectories $\{\xi_1, \xi_2, \dots\}$, in our case recorded for the assembly tasks. They are composed of state-action pairs $\xi_i = \{(s_0, a_0), (s_1, a_1), \dots, (s_N, a_N)\}$, with $s_i = (\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i) \in \mathcal{S}$ and $a_i = \boldsymbol{f}_i^h \in \mathcal{A}$ belonging to individual state and action spaces. Other than intending to learn a conventional mapping from state to action space $\mathcal{S} \rightarrow \mathcal{A}$, driven by the importance of iterative learning from past action-reaction characteristics, we rather intend to model and imitate the recordings jointly with a mapping $\pi : \mathcal{S}, \mathcal{A} \rightarrow \mathcal{A}$. During assembly, the obtained policy π shall predict plausible future actions for difficult situations, which can be passed to the robot control in form of $\boldsymbol{f}^h$ in Eq. (3.1).

Finally, this thesis proposes the following terms and definitions in the context of assembly for further sections:

Definition 1 ***Human Behavior****: Recorded trajectories $(s_{t-N}, a_{t-N}), \dots, (s_t, a_t)$ during expert demonstration, composed of both state $s = \boldsymbol{x}, \dot{\boldsymbol{x}}$ of the active assembly object and action $a = \boldsymbol{f}^h$ as the intended motor command.*

Definition 2 ***Skill****: A policy $\pi(\boldsymbol{x}, \dot{\boldsymbol{x}} \boldsymbol{f}^h)$, which models human behavior in form of a learned mapping $(s_{t-N}, a_{t-N}), \dots, (s_t, a_t) \rightarrow (a_{t+1})$ that observes past state-action relations and imitates plausible strategies for the next steps. This mapping is expected to encode error-correcting behavior against unexpected jamming, part tilting, and wedging with form and force closure.*

Definition 3 ***Strategy****: A sequence of actions $(a_t, a_{t+1}, \dots, a_{t+N})$ from the iteratively generated output of the skill π for a time horizon of few seconds. These short sequences of motor control are expected to solve problems of a short time horizon, such as wiggling to exploit stick-slip phenomena between parts in regions with low clearance or seeking haptic orientation through making controlled contact with edges, corners, faces.*

Definition 4 ***Skill Controller****: A controller through which a robotic system, usually an industrial manipulator, can assemble two parts, given an initial, nominal target trajectory $\boldsymbol{x}^d(t), \dot{\boldsymbol{x}}^d(t)$, and a final goal pose. The controller deploys an assembly skill π as an additional semantic component and uses the generated strategies $\boldsymbol{f}_t^h$ in a Cartesian control law for setpoint tracking.*

3.1.2. The Role of Simulation

Behavioral Cloning normally assumes that actions are observable. If that is not the case, the recent Behavioral Cloning from Observation (BCO) [151] can be used that learns from state-only trajectories. If the actions can be recorded, however, they provide a substantial source of information for task learning. While during human demonstrations on real hardware, see again Fig. 3.1, a tracking system could record $\boldsymbol{x}, \dot{\boldsymbol{x}}$, the action $\boldsymbol{f}^h$ is not straightforward to measure. Jäkel uses magnetic trackers and data gloves with finger-tip force sensors in his approach for learning household tasks [94]. This, however, has some drawbacks for learning assembly skills, conflicting with our requirements for industrial

context: Programming robotic applications is often done offline due to limited access to the platforms and the economic need to decrease down-times where possible. Moreover, assembly parts might be costly or not available, and making replicas or dummies means a significant engineering overhead. Instead, this thesis proposes to use simulation with a joystick-like input device to get access to the action space. Apart from meeting our requirements, simulation offers various benefits for machine learning, such as script-based preparation of environments and data collection at scale [152], pathing the way to a crowd-based generation of skill libraries. In contrast to original parts, Computer-Aided Design (CAD) data is commonly available for assembly.

Simulation has been used in combination with learning assembly skills from human performance. It provides the advantage of idealized environments for reproducible experiments and the fast acquisition of numerous trials and demonstrations. For assembly, early works used simulators to identify how human operators handle contacts and contact state transitions, seeking to derive symbolic programs [59], [60], simple alignment strategies [153], plan multi-step operations [154] or obtain sub-symbolic trajectory correction [111].

Aligned with those works, we use simulation as the primary environment for human skill extraction. In contrast, however, we omit the high-level construct of contact states and try to ground human assembly strategies directly in the cyclic interaction between intention $\boldsymbol{f}^h$ and outcome $\boldsymbol{x}, \dot{\boldsymbol{x}}$. In comparison to early days, simulators and physics engines have evolved significantly, becoming more and more powerful testing and prototyping environments for robotics research [155], [156].

In this thesis, simulation shall provide a suitable setup for extracting human behavior during assembly tasks. Interaction should be sufficiently intuitive and offer a game-like environment for users to play. Assembly in simulation should be easy to model but realistic enough to obtain concise recordings of human behavior.

3.1.3. The Role of Vision

Through closed-loop control with visual servoing, humans relate their strategies visually to the progress they make. When force feedback is not available, vision becomes indispensable. But as the only feedback channel, it's still sufficient for solving complex tasks in simulation [152]. This is mainly possible due to life-long learning and rich experience in manipulation, which we transfer across domains.

Imperfect localization on the robotic system and process-dependent issues, such as part occlusions during insertion, however, make it difficult to link assembly strategies to vision. In this thesis, we argue that observing the datastream of $\boldsymbol{x}, \dot{\boldsymbol{x}}$ is the low-level pendant to humans observing object motion directly and is, therefore, suitable to link assembly strategies to. The assumption is that skill learning can reveal underlying patterns in the raw signals $\boldsymbol{x}, \dot{\boldsymbol{x}}$, that are characteristic enough to generate meaningful strategies despite offsets in localization on the robot. In this regard, the velocity $\dot{\boldsymbol{x}}$ is of special importance. In later sections, we investigate and show its significance in an ablation study for determining its effect on learning success.

3.2. Skill Recording

The core of the concept builds on human-inspired skills and we need suitable methods to record them from teleoperation. To achieve this goal, Section 3.2.1 and Section 3.2.2 first describe our requirements for simulation environments and how low-clearance assembly can be realized with available physics engines. Section 3.2.3 then describes the process of teleoperating the simulator with a teach device and Section 3.2.4 analyses exemplary recordings. Insights from this analysis build the basis for the skill modeling of Section 3.3.

3.2.1. Simulation Environments

The primary goal of using a simulation environment in this thesis is to apply $\boldsymbol{f}^h$ to the active assembly parts and simulate the resulting motion $\boldsymbol{x}, \dot{\boldsymbol{x}}$. This task is accomplished with *forward dynamics* algorithms, of which implementation is a substantial part of any simulator for rigid multi-body physics. Through trial and error, operators steer the active objects towards the final assembly poses, solving friction-induced jamming and part wedging on the way.

There are usually various simulators available for robotics research. Open source, community-developed simulators include (non-exhaustive list): Gazebo [157], Klampt [158], Webbots [159], ARGoS [160], RobWork [146], and V-REP [161]. A more detailed discussion on frameworks can be found, e.g. in [162]. The increasingly popular becoming Mujoco [163] is a commercially available, closed-source simulator. In contrast to Computer-Aided Engineering (CAE) tools for dimensioning components, such as ADAMS[1], these robotics simulators usually target proof-of-concept implementations in robotics research. A trade-off is often made between speed and accuracy to favor applications in a real-time context.

For rigid body dynamics, the core of each simulator is a physics engine for numerically evaluating the equations of motion in a time-discretized manner. Widely used engines are e.g. ODE[2] [164], Bullet[3], and Nvidia's PhysX[4],

The following list of features is required to implement our concept of virtual assembly in simulation:

- Modeling of rigid bodies with mass and inertia
- Support of constrained motion simulation through contacts
- Representation of collision geometry with meshes. If the engine does not support non-convex polyhedra, an additional post-processing step from CAD to a feasible collision mesh might be necessary [165].
- Forward dynamics simulation with applying external forces
- Modeling of friction and damping in contact

In all these points, a game-like experience is more important than physical accuracy. Section 3.2.3 elaborates on the special case, in which choosing arbitrary, unrealistic masses

[1] https://www.mscsoftware.com/product/adams, accessed 8.8.2020
[2] https://www.ode.org/, accessed 7.8.2020
[3] https://github.com/bulletphysics/bullet3, accessed 7.8.2020
[4] https://github.com/NVIDIAGameWorks/PhysX, accessed 7.8.2020

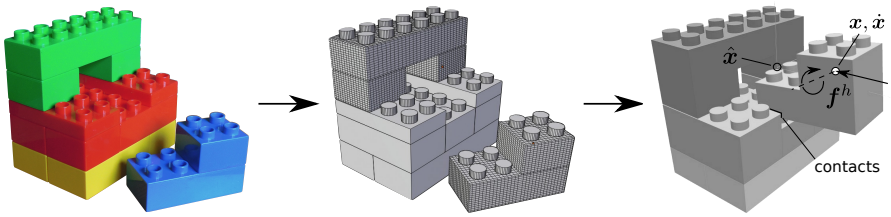


Figure 3.2.: Bringing the toy assembly into the simulation. From left to right: Real objects, mesh approximation with primitive shapes, simulated objects in Gazebo. Note that our approach only models the assembly objects for strategy learning in the task space. No robots are involved in these simulations. Although the simulator produces colored renderings, we mainly use grey-scaled images for improved illustration throughout further sections.

and inertias are tolerated, as long as the active objects move with *the right velocity* for people to naturally control the objects.

This thesis uses the Gazebo simulator with ODE as a physics engine. Gazebo has grown substantially over the years in features and performance. It's actively maintained and, through implementing an interface for the Robot Operating System (ROS) [17], [166], it's also widely used in the robotics community and connects well to various software modules, developed in this thesis. Fig. 3.2 shows the exemplary toy assembly's conversion from real to mesh and simulated objects in Gazebo.

3.2.2. Modeling Low-Clearance Assembly

Two peculiarities need consideration to implement our concept of skill extraction for low-clearance assembly: Contact stability and friction. The explanations target ODE as the underlying physics engine in Gazebo. Since ODE is widely used, this applies to various simulators.

Contact Stability

Numerically stable simulation is crucial for recording human behavior. Low-clearance assembly, however, is extremely challenging in that regard. One reason lies within the typical complexity of collision meshes. For non-convex, unstructured triangle meshes, contact discontinuities, and erroneously computed contact responses may lead to numerical instability, such as oscillations and jitter [158]. This is especially challenging for non-watertight meshes, but can be mitigated with Boundary Layer Expanded Meshes (BLEM) [158]. Our approach circumvents non-watertight, overly complex meshes by proposing to use low-polygon versions from CAD data, which originates from parametric, solid modeling of the assembly parts.

A more systematic problem lies within the constraint solver and the assembly task itself and occurs when contacts, constraining the motion of the objects in simulation, become redundant. We briefly discuss how ODE formulates constraints and how it mitigates this singularity problem.

We enrich ODE's explanations [164] with derivations from [167], adapted to the notation of this thesis. Using the Lagrange-Multiplier method, the constrained Newton-Euler

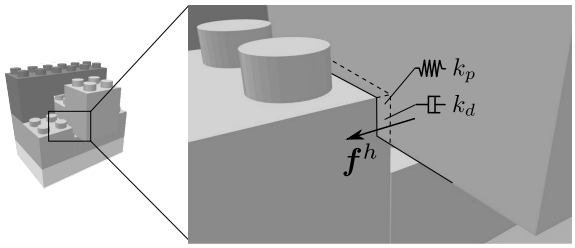


Figure 3.3.: Constraint relaxation with a spring-damper approximation through k_p and k_d. Tolerating a collision depth between surfaces improves numerical stability and removes oscillations for low-clearance assembly.

equations for a rigid body with mass matrix $\boldsymbol{M}$ are

$$\boldsymbol{M}\ddot{\boldsymbol{x}} = \boldsymbol{J}_c^T\boldsymbol{\lambda} + \boldsymbol{f}^{\text{ex}}. \tag{3.2}$$

The net force $\boldsymbol{f}^{\text{ex}}$ contains all inertial and Coriolis terms, grouped for simplicity. In combination with the constraint force vector $\boldsymbol{J}_c^T\boldsymbol{\lambda}$ with the unknown Lagrange Multipliers $\boldsymbol{\lambda}$, the system must comply with motion constraints, which are formulated on velocity level as a set of equations of the form $\boldsymbol{J}_c\dot{\boldsymbol{x}} = \boldsymbol{c}$, using the constraint Jacobian $\boldsymbol{J}_c$. A peculiarity in ODE is to add a relaxation term for *constraint force mixing* through a diagonal, square matrix $\boldsymbol{C}_{FM}$ to this set:

$$\boldsymbol{J}_c\dot{\boldsymbol{x}} = \boldsymbol{c} + \boldsymbol{C}_{FM}\boldsymbol{\lambda}. \tag{3.3}$$

This allows the constraints $\boldsymbol{c}$ to be violated proportional to $\boldsymbol{\lambda}$ which should enforce them. The effect of this is apparent after two reformulations. Solving Eq. (3.2) for accelerations and performing one explicit time integration with step width h leads to

$$\dot{\boldsymbol{x}} = \boldsymbol{M}^{-1}\boldsymbol{J}_c^T h\boldsymbol{\lambda} + \boldsymbol{M}^{-1}\boldsymbol{f}^{\text{ex}}h, \tag{3.4}$$

which describes the object's motion in each cycle. Setting Eq. (3.4) into Eq. (3.3) leads to

$$\left(\boldsymbol{J}_c\boldsymbol{M}^{-1}\boldsymbol{J}_c^T + \frac{1}{h}\boldsymbol{C}_{FM}\right)\boldsymbol{\lambda} = \frac{\boldsymbol{c}}{h} - \boldsymbol{J}_c\boldsymbol{M}^{-1}\boldsymbol{f}^{\text{ex}}h, \tag{3.5}$$

which is solved as a Linear Complementary Problem (LCP) in each step. Having solved Eq. (3.5) for $\boldsymbol{\lambda}$, the object's motion is given by Eq. (3.4).

Without $\boldsymbol{C}_{FM}$ the problem's ill-conditioning depends on the rank of $\boldsymbol{J}_c$. The mass matrix $\boldsymbol{M}$ is positive definite and thus invertible. For low-clearance assembly in simulation, the meshes collide at various points simultaneously. This is more pronounced, the more geometry constraints the active object's motion, which is classically the case towards the end of the assembly. In these frequent cases, $\boldsymbol{J}_c$ becomes rank-deficient through redundant or conflicting constraints [167], so that $\boldsymbol{J}_c\boldsymbol{M}^{-1}\boldsymbol{J}_c^T$ from Eq. (3.5) becomes singular, leading to numerical instability. Constraint force mixing avoids this at the cost of allowing mesh interpenetration. For identical spring-damper behavior in all dimensions, $\boldsymbol{C}_{FM}$ can be built with stiffness k_p and damping k_d according to [164]:

$$\boldsymbol{C}_{FM} = \text{diag}\left(\frac{1}{hk_p + k_d}\right). \tag{3.6}$$

Fig. 3.3 illustrates both constants for the exemplary toy assembly. In conclusion, surface interpenetration needs to be tolerated with a certain margin to obtain numerical stability. This degrades realism but avoids jitter in simulation, which would corrupt velocity measurements $\dot{\boldsymbol{x}}$ and lead to discontinuous curves in $\boldsymbol{x}$. For CAD data with low nominal clearance in the final assembly, e.g. plastic housing components or plug-in connectors, this modeling technique is indispensable for numerically stable simulation.

Our concept of obtaining assembly strategies with primitives supports this degraded realism. During the demonstration, steering the active object with $\boldsymbol{f}^h$ in this spring-damper boundary layer *appears* highly viscous and stiction dominates this contact, hindering the users to advance. Intuitively, the reaction is, therefore, to avoid unnecessary interpenetration during demonstrations as much as possible. A high stiffness k_p will constantly *enforce* to work in the nominal clearance.

Friction and Modeling Accuracy

Most physics engines approximate friction with the Coulomb model. We use ODE with a linearized, pyramidal approximation of the friction cone for improved performance. More realistic implementations exist, e.g. [168], but our approach does not require this detail. Friction does indeed affect the strategies that users will apply in simulation. To better imagine this correlation, consider Fig. 3.4. With little friction, simple goal-directed pushing is a valid strategy. Higher friction favors force closure, requiring strategies to contain wiggling. Friction is in general difficult to estimate for assembly tasks offline. CAD data with material combinations might be accessible for some assembly tasks, but this thesis' approach shall also encompass the cases where these parameters are not available. Hence our overall goal is to obtain robust strategies by overestimating friction. We thus make simulation notably harder than the real assembly.

Obtaining robust strategies is strongly interconnected with what's often referred to as the *reality gap*. The problem lies in the transferability of skills π to real-world settings, that are solely learned on synthetic data, generated in simulators. For robotic manipulation in general and assembly in particular, a multitude of closely-matching parameters are usually required to assure task success, such as micro-slip friction, stiction, restitution, non-rigid contact deformations. We tackle this challenge by seeking *normed* strategies that make human assembly skills. The idea is to learn what is qualitatively helpful albeit being created in simulation. What we do need is a later calibration on the real hardware by globally scaling the learned strategies before sending them as reference setpoints to robot force control. The hypothesis underlying our approach is that human behavior, even in this limited environment, is a sufficiently concise source for learning robust assembly strategies. Chapter 5 will evaluate the validity of these assumptions on various real-world assemblies.

As a conclusion for modeling accuracy, there are some parallels to constrained motion planning. We also emphasize constraint modeling and contact physics, albeit with lower details and realism of modeling. Both would be crucial for kinodynamic motion planning, mainly due to the open-loop execution. In contrast, this thesis' concept builds on reactive strategies, learned from a human planner. We are thus able to use simpler simulation environments and bridge the gap of simulative shortcomings with the reactiveness of human-inspired strategies.

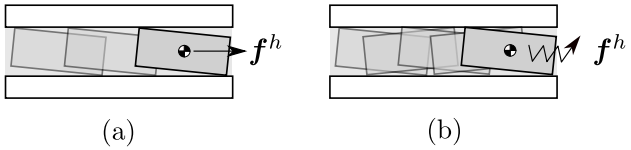


Figure 3.4.: Illustration of the friction effect on strategies. (a) Low friction supports pushing. (b) High friction supports wiggling.

3.2.3. Recording Human Behavior

Coordinate Systems and the Correspondence Problem

When considering skill learning in the context of Learning from Demonstration, the *correspondence problem* [169] is of particular interest. In robotics, it mostly refers to the challenge of transferring learned skills from the teacher to the robotic system that should execute the task. Especially learning from observation techniques are prone to this issue [96]. At the core of the problem lies a non-trivial mapping between different action spaces of teacher and robot. Jäkel [94] achieves a solution for this mapping through constrained motion planning. An important step is relaxing constraints that are only associated with human morphology without offering semantic content for the task. His approach is mostly independent of the robot morphology, considering basic requirements regarding the degrees of freedom. In our scenario, the human is likewise involved as the expert in the demonstration. But through the provided interface to simulation, the teacher is forced to demonstrate in task space instead of the configuration space of his own joints. This constitutes an alternative solution to the correspondence problem and is characteristic of learning in task space via teleoperation [96]. Chapter 4 considers the actual transfer to robots in detail.

In contrast to many household tasks, such as pouring a drink [94], in assembly, the exact orientations of the parts do matter. We still achieve workspace independence of the skills through formulating all quantities of interest with respect to assembly-relative frames. By our convention, the assembly origin $\mathcal{O}$ shall be placed in the passive object in such a way that it coincides with the active object's reference frame $\mathcal{A}$ after successful assembly. Fig. 3.5 illustrates this convention. We choose $\boldsymbol{x} = [x, y, z, q_w, q_x, q_y, q_z]^T \in \mathbb{R}^7$ for representing a 3D pose through position components and the orientation quaternion. After a successful assembly, $\boldsymbol{x} = [0, 0, 0, 1, 0, 0, 0]^T$.

$\boldsymbol{f}^h$ is applied to the active object in $\mathcal{A}_o$. Maintaining global orientation for forces and torques provides a consistent feeling of steering the object in the simulator. The active object's pose and velocity $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ are recorded with respect to the assembly origin $\mathcal{O}$. When setting up a simulation environment for an assembly task, specifying the center of mass of the active object is a design choice. The other frames are derived from our convention. In particular, this center of mass need not correspond to the real one. It's more a choice of where the objects are most intuitively manipulated through $\boldsymbol{f}^h$. A good choice is to put this close to where one would normally grasp the active object.

Mass, Inertia and Execution Speed

Modeling assembly in simulation might suggest matching the real hardware parameters as close as possible. Our concept, however, circumvents this engineering effort. CAD

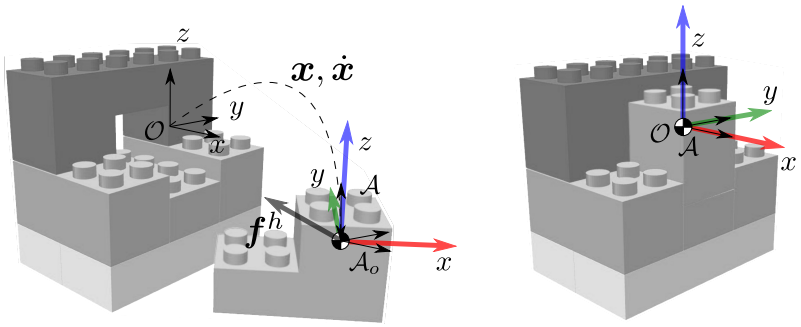


Figure 3.5.: Placement of reference frames for assembly tasks in simulation according to our convention. $\mathcal{A}$ is the RGB-colored coordinate system attached to the active object and defining its center of mass. $\mathcal{O}$ denotes the assembly target, fixed with the passive object. $\mathcal{A}_o$ is linked to $\mathcal{A}$ but keeps the global orientation of $\mathcal{O}$. After successful assembly, all frames must coincide.

data might include estimates of mass and moments of inertia, but they are not required for our concept. Similar to using a coarse and conservative, i.e. too high estimate of friction, our concept proposes to use artificial masses and moments of inertia with the goal of achieving comfort in control, rather than physical consistency. The primary goal of the assembly simulation is to model adequate responsiveness for users, steering objects with $\boldsymbol{f}^h$. Towards this end, we use zero-gravity environments, i.e. we set the gravity constant in the physics engine identically to zero. When $\boldsymbol{f}^h = \mathbf{0}$, everything should stay in place.

Mass and moment of inertia will determine the object's acceleration according to Eq. (3.2). Our concept's goal is to have a linear response to $\boldsymbol{f}^h$ so that it's easy to steer the active objects. For this aspect, we combine low apparent inertia with high velocity-proportional damping according to

$$\begin{pmatrix} \boldsymbol{I}^{3\times3} d_{\text{lin}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}^{3\times3} d_{\text{rot}} \end{pmatrix} \dot{\boldsymbol{x}} + \boldsymbol{f}^h = \mathbf{0} \tag{3.7}$$

in which $\dot{\boldsymbol{x}} = [\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z,]^T \in \mathbb{R}^6$ is the active part's vector of linear and angular velocity, d_{lin}, d_{rot} are linear and rotary damping constants, and $\boldsymbol{f}^h = [f_x, f_y, f_z, t_x, t_y, t_z]^T \in \mathbb{R}^6$ is the wrench of forces and torques applied by the user via the teach device. We add both the damping term and $\boldsymbol{f}^h$ to the combined vector of external forces and torques $\boldsymbol{f}^{ex}$ from Eq. (3.2) to achieve the velocity-proportional, linearized control in the simulator.

The overall execution speed, governed by d_{lin}, d_{rot}, and the amplitude of $\boldsymbol{f}^h$, is again a design choice. In contrast, in the real setup, the robot force control sets the speed of manipulation due to stability requirements. In general, the stiffer the environment and the lower the closed-loop control rate, the lower the tolerable execution speeds for assembly. Section 3.3 explains how a policy π is learned from this simulated data and how sequences of discrete states scale to different execution speeds.

The Teach Device

For teleoperation, haptic feedback during demonstrations could lead to a more immersive experience. Our concept doesn't collide with haptic devices, but it's general enough to not require them. Additionally, we propose to focus solely on visual feedback instead,

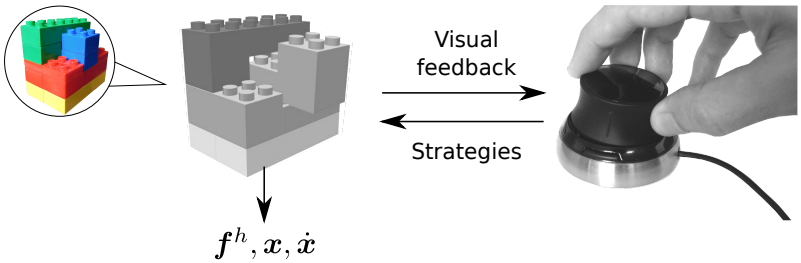


Figure 3.6.: Behavior recording with the simulation environment. An operator steers the active object to assemble two parts using a 3D controller. This teach device maps translational and rotary displacements to the 6-dimensional control command $\boldsymbol{f}^h$. The recorded behavior is stored as a dataset over repetitive trials.

which we believe has some advantages. If used in our setting, the haptic device would render the physics engine's constraint forces $\boldsymbol{J}_c^T \boldsymbol{\lambda}$ from Eq. (3.2) for the operator. In the central nervous system of man, haptic feedback overrules geometric feedback [170]. This, however, would emphasize the need for the correctness and physical plausibility of these constraint forces to generate a natural feeling and thus not negatively impact the cognitive skills, associated with geometry. Given the issues of numerical stability during low-clearance assembly, this appears to be a significant challenge. Unrealistic haptic feedback could distract during the task. Using only vision helps to prevent operators from learning the quirks of the simulator instead of objectively solving the assembly task.

Due to repetitive demonstrations, operators themselves perform visuomotor learning in the simulator. Research in human motor learning indicates that immediate visual feedback is crucial for learning tasks with high functional complexity [171]. This underlines visual feedback as the most powerful modality for extracting the cognitive part behind assembly skills and motivates its central role in our approach.

Fig. 3.6 shows the recording process of human behavior in the simulator with the teach device. We use a 3Dconnexion™ 3D controller as a joystick to intuitively command in six dimensions: three translations and three rotations. Through modeling a linear, six-dimensional stiffness, we map displacement signals from the resting position of the device to obtain the combined force-torque vector $\boldsymbol{f}^h$. The exact parameterization is again a design choice but will determine the nominal magnitude of the strategies, i.e. it will set the *default strength* of strategies learned from the recorded data. As an example, for the toy assembly in the simulator, we initially parameterize $\boldsymbol{f}^h$ with its force components being bounded by $\pm 34\,\mathrm{N}$ and the torque components by $\pm 3.4\,\mathrm{Nm}$. By our definition, strategies are sequences of combined force-torque data points $\boldsymbol{f}_t^h, \boldsymbol{f}_{t+1}^h, \ldots, \boldsymbol{f}_{t+N}^h$. Since they are vector quantities, we can re-scale the whole sequence in magnitude without losing its characteristics. This allows an adaptation of the strategies to meet specific requirements and not exceed thresholds, posed by the real hardware during force control.

The benefits of using a simple joystick-like 3D controller are manifold: Mechanical engineers are used to such devices during development in CAD for intuitive 3D navigation and inspection of parts on the screen. Online robot programmers are also familiar with this concept from their robot programming interfaces, which are commonly hand-held devices with similar joysticks for Cartesian jogging. Although not a scientific focus, using a 3D controller in our concept thus benefits an integration into industrial applications.

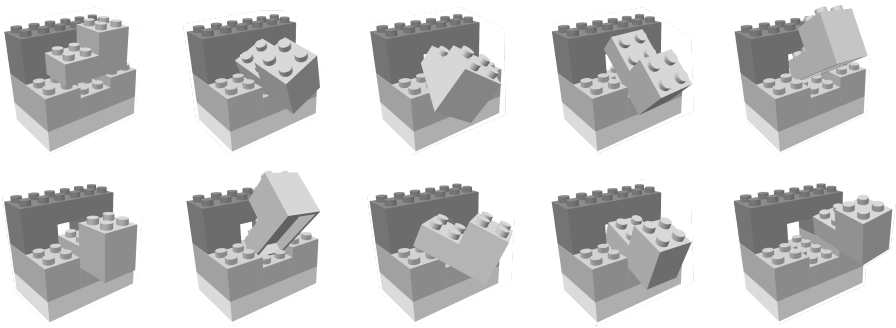


Figure 3.7.: Challenging start configurations in the simulator for the toy assembly task.

Skill Extraction

Through imagination, humans should be able to transfer their manipulation skills and experience to virtual tasks. In detail, we are interested in what strategies demonstrators associate with geometric constellations of the assembly parts. We thus create challenging configurations in the simulator to trigger these strategies. Fig. 3.7 shows an excerpt of challenging starting positions. During data acquisition for an assembly task with our concept, various starting positions are sampled randomly to cover a big range of possible jamming and tilting effects, so that the strategies recorded provide answers for difficult deadlocks that might appear during the real assembly.

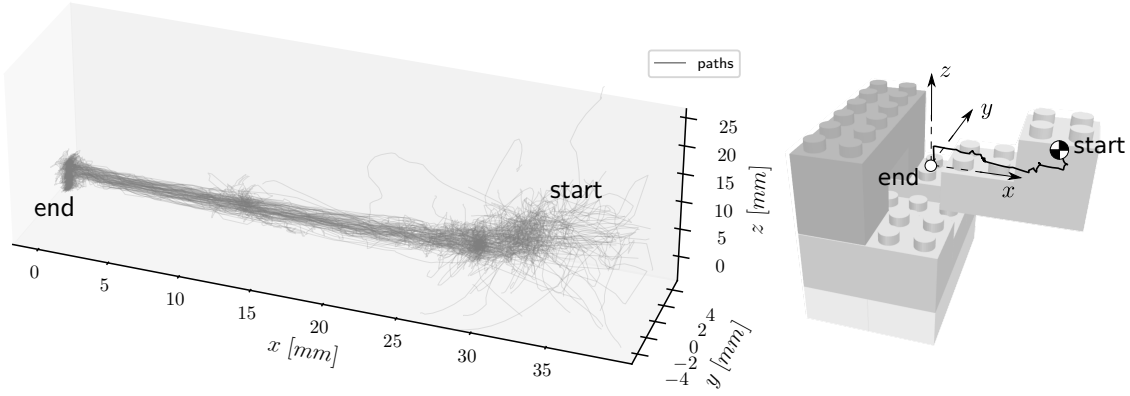
3.2.4. Data Analysis

The goal of this section is to prepare a basis for algorithmic choices for skill modeling and learning. We use the toy assembly from previous sections as a representative and illustrative example of a low-clearance assembly with friction and a multi-axis insertion process. We directly plot and analyze recordings made in the simulator. To create a sufficient basis for statistical analysis, we use the data from 100 demonstrations as shown in Fig. 3.8. The demonstrations started at partially random poses in close proximity of making the first contact and ended at $x, y, z = 0$.

For illustration purposes, we use the Euclidean distance to the goal as one-dimensional reference to illustrate the results in 2d plots. After describing a funnel at the beginning of insertion, the low clearance of the parts makes this measure sufficiently unique towards the goal. The Strategies $\boldsymbol{f}^h$ are of particular interest since they represent the action space that we want to learn from human behavior. We additionally analyze the variances of this action space and illustrate regions of trial and error during manipulation. Finally, displaying features over time gives insights into sequential characteristics.

Strategies

By our definition, strategies describe the actions $\boldsymbol{f}^h$ over time. The goal here is to analyze how this action space looks from a time-less perspective along the insertion. Fig. 3.9 shows the 100 demonstrations along the distance-to-goal reference. Each band plot shows the spectrum of recorded forces and torques, bounded by min and max values. The thicker this band, the greater the ambiguity among individual demonstrations. This is especially pronounced in the torques t_x, t_z with an almost constant thickness. In compar-



(a) (b)

Figure 3.8.: Setup for data analysis: (a) The xyz-traces of 100 assembly demonstrations in simulation with varying start positions. The traces show the active object's path until reaching the goal position at $x, y, z = 0$. (b) One of such start constellations in the simulator with an overlay of the recorded path. During each demonstration, the tuple $(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{f}^h)$ was recorded at $100\,\mathrm{Hz}$.

ison, the vertical force f_z shows a region between $25\,\mathrm{mm}$ and $20\,\mathrm{mm}$ to the goal, where strategies seem to avoid excessive forces. Additionally, a single demonstration reveals trial and error at difficult spots in the assembly, e.g. at $5\,\mathrm{mm}$ to the goal. Note how learning the mean would completely lose the individual characteristics of these strategies.

Variances

Fig. 3.10 plots the variances of $\boldsymbol{f}^h$ along the goal distance. The curves are noisy in general but expose regions of low and high variance. For example, after one-third of the distance, f_y shows a relatively low variance and indicates that the strategies contained little lateral control. In comparison, the vertical force component f_z reveals high variances at $15\,\mathrm{mm}$ and $5\,\mathrm{mm}$ that could hint at difficult spots with random trial and error. In between these peaks, low variances indicate consistency among demonstrations to some extend. The torque components t_x and t_z are characterized by high variance in general. It would be difficult to learn these strategies using only position information. From a probabilistic model perspective, the high fluctuations in all subplots require fine-grained distributions to accurately describe this dataset.

Feature Histograms

Fig. 3.11 shows histograms of $\boldsymbol{f}^h$ and $\dot{\boldsymbol{x}}$. Darker areas mean an accumulation of recorded data for specific regions. This gives an impression of where it was difficult to make progress in the assembly and took longer to advance. Although the spots are task-specific, the forms of the gradients reveal an inverse learning problem: Using only x, y, z-positions to learn $\boldsymbol{f}^h$ and $\dot{\boldsymbol{x}}$ equals a multivariate decision problem. This is also the case for learning behavior in the almost blank spaces in between.

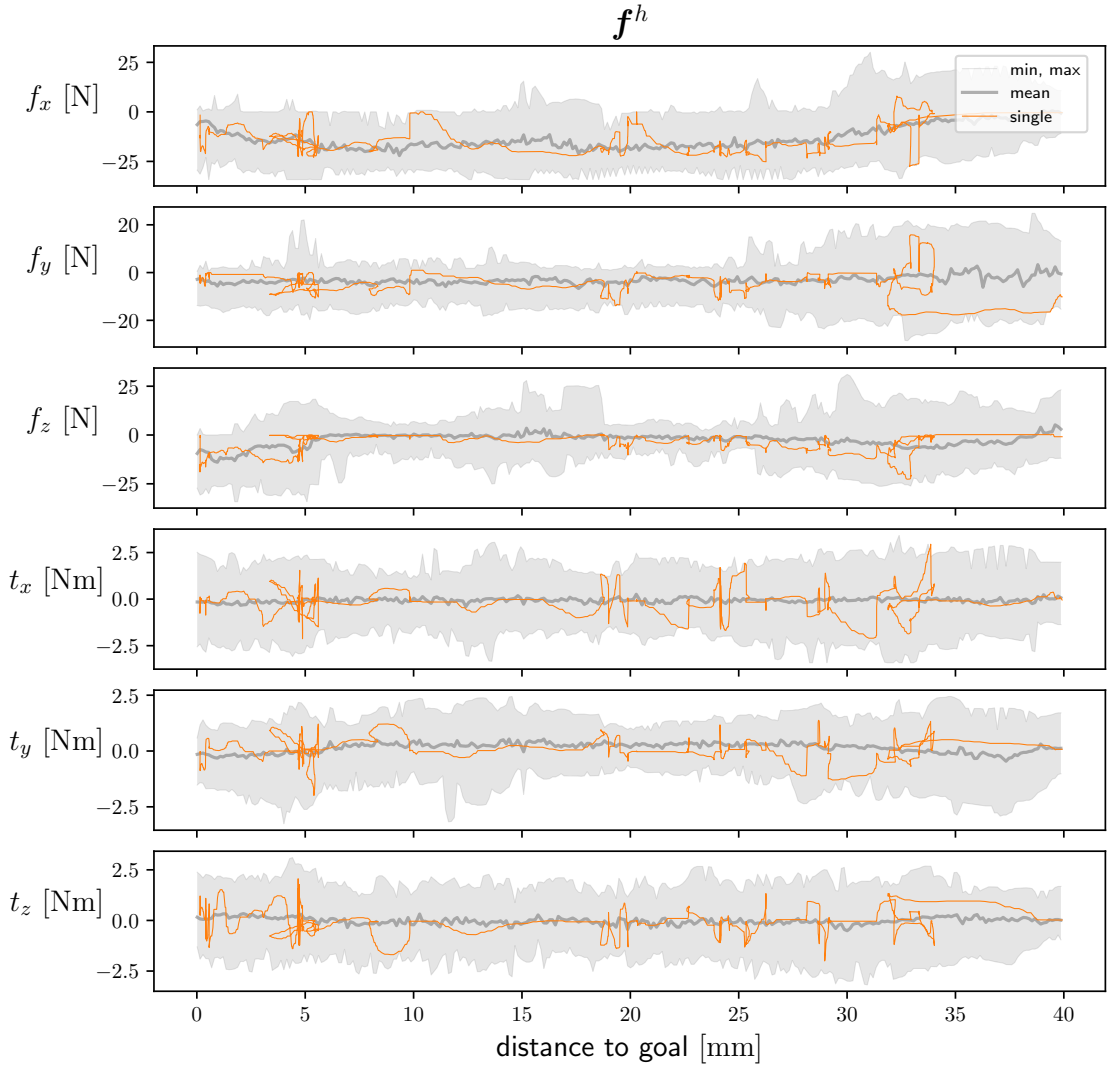


Figure 3.9.: Band plot of the control wrench $\boldsymbol{f}^h$ applied during 100 demonstrations. For low clearance tasks, the distance to the goal provides a mostly unambiguous, one-dimensional reference and allows an investigation of different geometric assembly constellations. We truncate the data after $40\,\mathrm{mm}$ of distance to the goal before which free-motion without contact occurs (right end of the plot). To obtain the plots, we use statistical data binning to group the continuous values of all demonstrations into vertical bins for the ordinate axis. We then compute the min, max, and mean values within those bins and plot them as continuous curves. In addition, a single demonstration shows exemplary strategies during assembly.

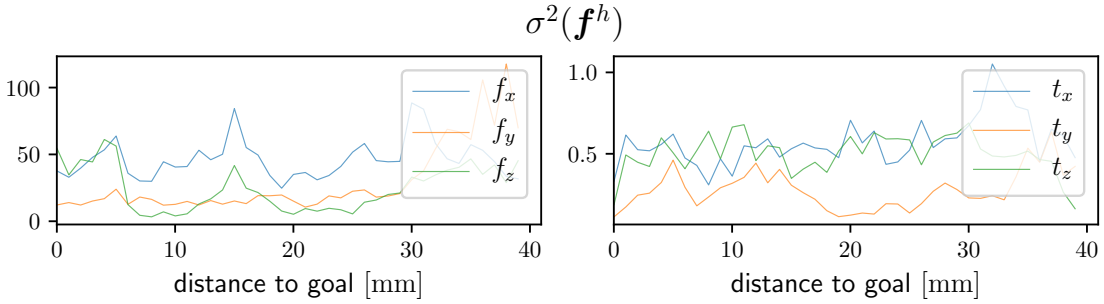


Figure 3.10.: Variances of the control wrench $\boldsymbol{f}^h$ towards the goal. We obtained the variances with statistical binning, analog to Fig. 3.9. For this plot, we grouped the data into 40 bins from start to goal and computed the variance of a normal distribution for each bin.

Features over Time

Fig. 3.12 shows five of the 100 demonstrations. Each component reveals patterns over time and the demonstrations are similar, except for scale and offset. Various effects are visible: The torque t_z shows wiggling in form of alternations, which supports assumptions on trial-and-error from previous investigations. q_z and ω_z show the resulting motion about that axis. Their amplitudes decrease towards the end where clearance is significantly lower. For this section, the force f_z shows pushing profiles, whose characteristics start at around $8\,\mathrm{s}$ for the fastest demonstration and at around $13\,\mathrm{s}$ for the slowest demonstration. In contrast to assuming purely geometry-related behavior, including sequential information constitutes a promising base for learning patterns in our dataset.

Conclusions

The exemplary dataset of 100 assembly demonstrations in simulation showed high variance between individual recordings for our geometric measure. Low clearance and high friction require repetitive trial and error in specific regions and are representative of challenging insertion. Averaging this search-like behavior is not feasible, because it loses the individual strategies. Although seemingly random towards the goal, the recorded behavior showed patterns in sequences and thus motivates a temporal probabilistic model for skill learning.

3.3. Skill Modeling

Deep neural networks [172] are powerful function approximators for data-driven approaches. In this thesis, we use a specially combined neural network as a regression model for human-inspired assembly skills. As elaborated in the previous section, two basic components will be particularly important:

1. A *memory* component that is capable of making use of past, sequential behavior and patterns in data.

2. A *probabilistic* component that learns to reproduce human behavior's seemingly random and arbitrary characteristics.

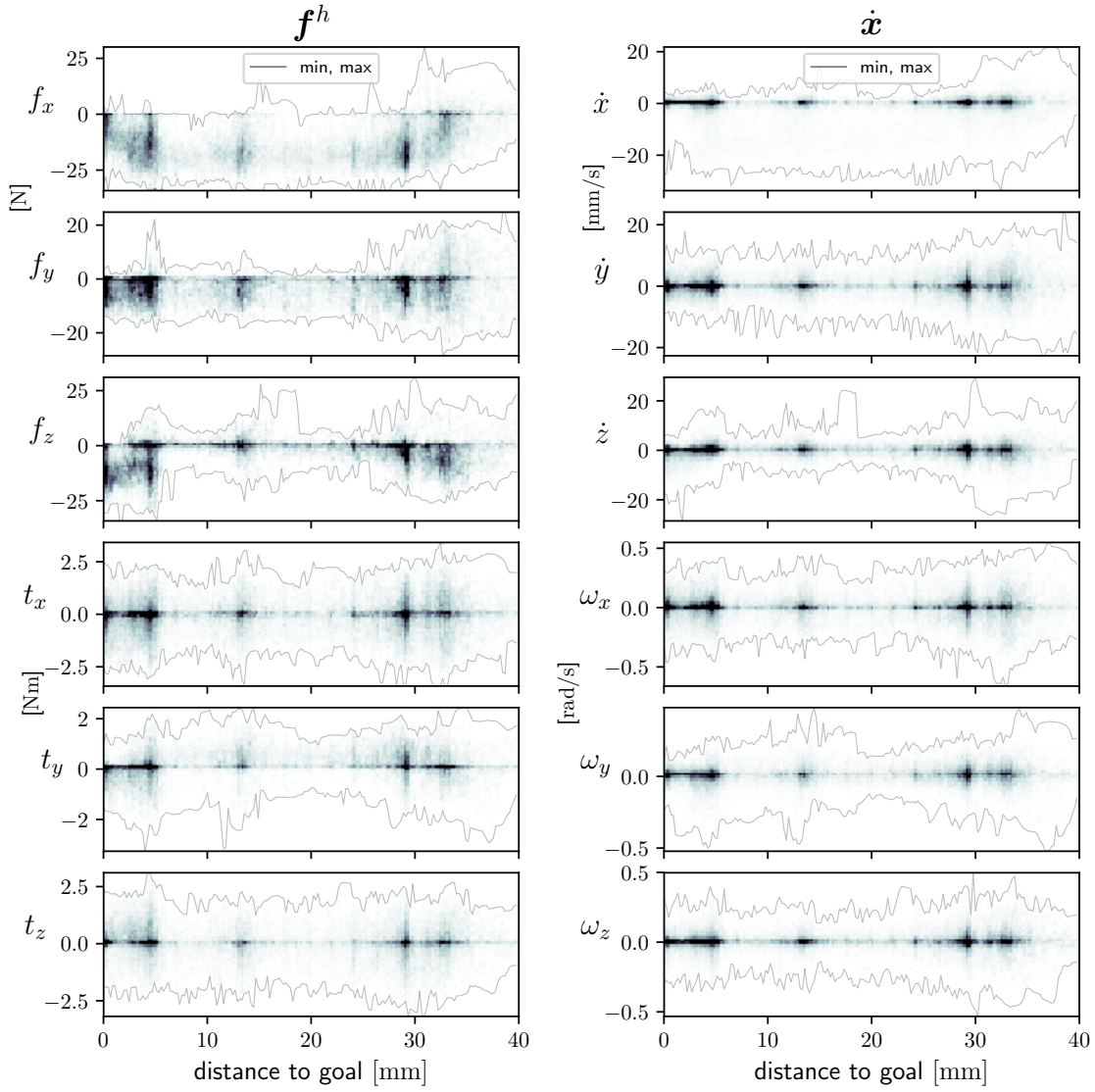


Figure 3.11.: 2d-histograms of all components of $\boldsymbol{f}^h$ and $\dot{\boldsymbol{x}}$. The color gradient shows the qualitative occurrence of discretized values for forces f_x, f_y, f_z, torques t_x, t_y, t_z, linear velocities $\dot{x}, \dot{y}, \dot{z}$ and angular velocities $\omega_x, \omega_y, \omega_z$ in a 180×60 grid. Darker colors represent higher occurrences of the ordinate values for specific regions. The min and max curves of Fig. 3.9 are added for a better illustration of boundaries.

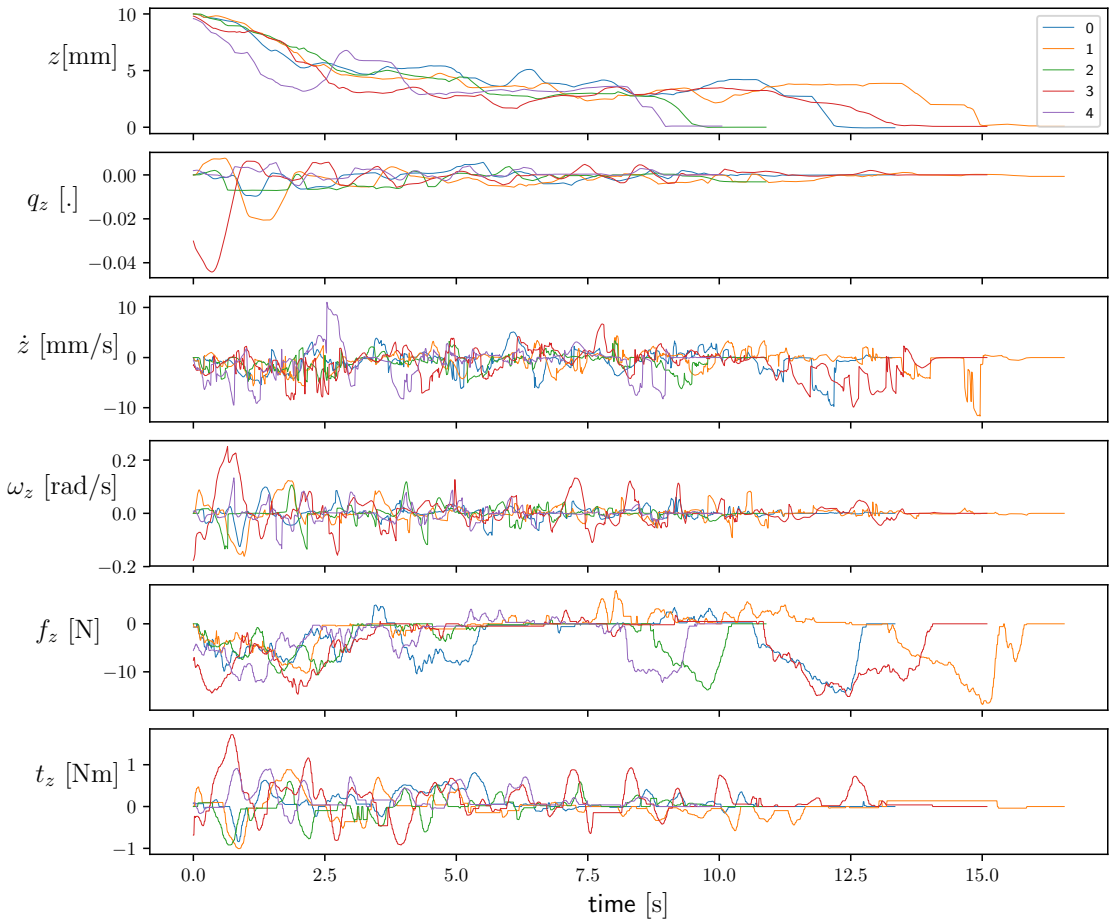


Figure 3.12.: Five exemplary demonstrations for a subset of features. These demonstrations had equal starting poses of $\boldsymbol{x} = [35\,\mathrm{mm}, 0, 10\,\mathrm{mm}, 0, 0, 0, 1]$ for better comparison. The vertical position z illustrates the assembly's progress. Reaching zero indicates the end of the assembly. Their durations were $10.0\,\mathrm{s}$, $10.9\,\mathrm{s}$, $13.3\,\mathrm{s}$, $15.1\,\mathrm{s}$, and $16.6\,\mathrm{s}$.

Section 3.3.1 and Section 3.3.2 describe these components in detail and Section 3.3.3 describes the combined, final neural network model for learning human-inspired assembly skills.

3.3.1. Learning Patterns in Sequences

The first component of our skill model has the task of making use of the sequence character of assembly behavior. By our definition, we consider behavior as sequences of state-action pairs (s_i, a_i), whose actions form valuable strategies for assembly. Looking at sequences instead of at individual, disjoint states is appealing for a simple reason: Having observed the evolution of states across time gives us much more semantic background for our next actions. The data analysis from the previous section implied that recorded behavior is in itself consistent, but appears random when observed without a timely context.

For sequence learning, the class of Recurrent Neural Networks (RNNs) has gained enormous popularity. In contrast to feedforward neural networks, the individual units possess recurrent connections, which make them cyclically working, dynamic systems. This seemingly minor difference allows information to persist in the network, such that long-passed inputs can in theory influence any future decisions. For learning correlations in time, this is a crucial feat. One of the biggest, early challenges was solving the vanishing and exploding gradients problem [173], [174], that limited practical applications of canonical RNNs to few time steps [175]. This has mainly been solved by Long Short-Term Memory (LSTM) [176], which has become a powerful and very popular component in various sequence-related applications, such as generating hand-written texts [177] or performing language translation [178]. Other network architectures for sequence modeling include e.g. the Gated Recurrent Unit (GRU) [179] or time window-based feedforward nets [180]. While the first one is a simplification with similar performance, the second one models the receptive field with dilated causal convolutions instead of with recurrence. This idea is not new and has been used in earlier work [181]. But being a feedforward network, an inherent advantage of this approach is simpler training in comparison to RNNs. A recent study [182] provides an empirical comparison between canonical LSTMs and such Temporal Convolution Networks (TCN) and suggests reconsidering TCPs as the default solution for sequence learning.

There are, however, two reasons why this thesis uses LSTMs for sequence modeling. The first reason is that TCPs require to pass the raw history of input sequence length to the network for prediction. In a real-time context, this constitutes an unnecessary performance overhead. In contrast, RNNs need only the next input and an internal state in latent space keeps track of the history. The second reason for using recurrence for modeling assembly skills is that TCPs also require to fix the network's perceptive field to concrete sequence lengths. This makes them less flexible during applications, where RNNs can have adaptive up to theoretically infinite memory. We make use of this flexibility during the investigation of the effect of memory length for skill learning in Section 5.1.4

Advances continue for LSTMs, e.g. [183], and RNNs in general [184]. This thesis aims to show that the success of learning human-inspired assembly skills does not depend on the latest improvements in the method. But rather is already achieved through carefully chosen, canonical models. We, therefore, use the well-established version from Gers et al [175] as a central component in this thesis for modeling human behavior during assembly tasks. Through being a basic, non-specialized component, we expect alternatives, such

as [179] to work comparatively well with our approach.

Mapping Approaches

Before explaining the details of our LSTM-based network, this paragraph discusses input-output relations for sequence modeling in general.

One-to-One: $(s_t, a_t) \rightarrow (a_{t+1})$
This could be considered a corner case with the input and output sequence length of a single time step. Used on the recorded data, this model will be problematic: The resulting policy π models strategies as a static field of forces and torques, with the conflicting goals of not overfitting while capturing fine-grained patterns. When including probabilistic components for stochastic learning, such as in Variational Autoencoders (VAEs) [185], this model assumes that data are independently and identically distributed (i.i.d.), which is not the case due to correlations in time. It will be difficult for the model to learn this ambiguity, resulting in a high variance of the distributions. The consequence for robot control is either uncorrelated, fluctuating draws through sampling or averaged strategies through least-squares regression.

One-to-Many approach: $(s_t, a_t) \rightarrow (a_{t+1}), \ldots, (a_{t+N})$
This model makes predictions based on a single input. Motivated by an application of image caption generation [186] and transferred to robotics, we investigated this model in our earlier work for assembly [187]. Instead of using memory for observing the past, the sequence-generating LSTM network used memory to predict coherent, future strategies, which were sent open-loop to the robot control to overcome deadlocks of the one-to-one model. While this worked reasonably well also with uncertainty for the case considered, this approach has limits when clearance is considerably smaller and a lot of strategies are applied in relatively small geometric regions. Low clearance leads to ambiguity of the recorded behavior, which gets lost in deterministic mappings. As a consequence, the learned average loses the random-appearing character of strategies and fails in friction-dominated, stick-slip tasks. Section 3.3.2 describes how this thesis instead tackles this issue with a probabilistic output layer.

Many-to-Many: $(s_{t-N}, a_{t-N}), \ldots, (s_t, a_t) \rightarrow (a_{t+1}), \ldots, (a_{t+N})$
This is a common architecture for sequence translation models, such as [178]. The core idea is to use an encoder for the input sequence and decode an output sequence from this compact latent space. While this could be a promising model for assembly skills, having an output sequence like the one-to-many models means dead times on the robot control. For the time of execution, the strategies are open-loop. For short output sequences, however, this could still be a feasible solution

Many-to-One approach: $(s_{t-N}, a_{t-N}), \ldots, (s_t, a_t) \rightarrow (a_{t+1})$
Taking the idea of using an encoder for learning a compact representation of input sequences into latent space, this model generates an immediate control signal for real-time robot control.

Basing on own research of all models, this thesis uses the many-to-one type to model assembly skills. To keep equations in the next sections short, we introduce two abbrevia-

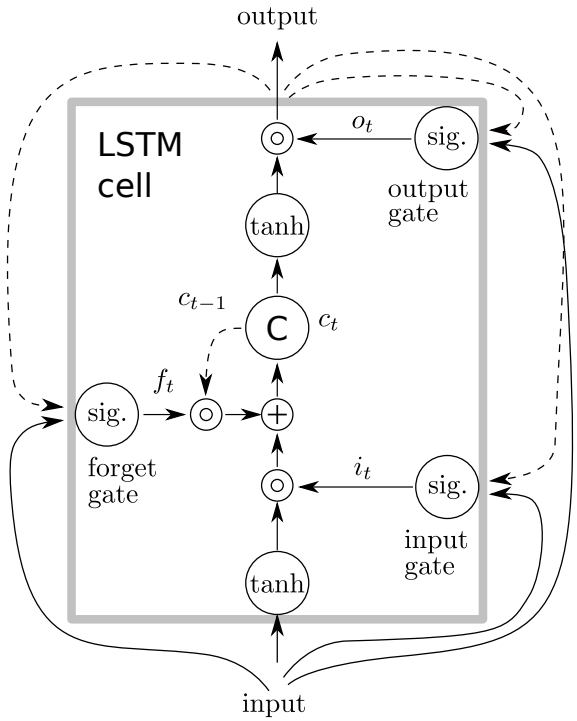


Figure 3.13.: Long Short-Term Memory (LSTM) cell, adapted from [186]. The dashed lines represent recurrent connections. Characteristic for this cell are three gated, multiplicative operations that enable to effectively memorize and forget information stored in the internal cell state c_t.

tions:

$$\hat{\boldsymbol{x}}_t = (s_t, a_t) \qquad = \left[\boldsymbol{x}_t^T, \dot{\boldsymbol{x}}_t^T, \boldsymbol{f}_t^{h\,T}\right]^T \tag{3.8}$$

$$\hat{\boldsymbol{y}}_t = (a_{t+1}) \qquad = \boldsymbol{f}_{t+1}^h. \tag{3.9}$$

The right-hand side represents column vectors for multiplication with weight matrices. In the terminology of supervised learning, $\hat{\boldsymbol{y}}_t$ can be considered as the *label*, i.e. the *correct* output for a given input $\hat{\boldsymbol{x}}_t$. This aspect is used in Section 3.3.2 by formulating the learning problem of human behavior through learning to estimate high conditional probability densities for these labels.

Long Short-Term Memory (LSTM)

We use the LSTM to observe and encode patterns in behavioral sequences into a concise representation. We will later combine this with a probabilistic output layer. Using the encoded representation from the LSTM, this final layer shall be able to iteratively predict future strategies with high confidence.

The implementation we use[5] [188] bases on [176] and [175]. Fig. 3.13 shows the individual LSTM cell. Crucial for memorizing and forgetting are gated operations that can easily add and subtract information from the cell state c_t. In its canonical form [176],

[5] https://github.com/tensorflow/tensorflow/blob/r2.0/tensorflow/python/keras/layers/recurrent.py

the LSTM's cell state had a recurrent self-connection with weight 1.0 that created a "constant error carousel", generating a linear error backflow, protected by input and output gates [175]. In addition to memorizing by default, however, a practical mechanism for occasional resetting was required to avoid keeping useless information. By introducing a forget gate, Gers et al [175] enabled the LSTM to gradually reset the cell state's obsolete information. In combination, both the input and forget gate operations learn to keep and drop valuable information in and from the cell state. For learning human assembly skills, this enables us to use the data without explicit start-end annotations. For practical applications, various of these cells are combined into an LSTM-*layer*, allowing to describe the information flow in the forward pass with the following matrix-based set of equations (for m individual LSTM units from Fig. 3.13):

$$\begin{aligned}
\boldsymbol{f}_t &= \text{sigmoid}(\boldsymbol{W}_f \hat{\boldsymbol{x}}_t + \boldsymbol{b}_f + \boldsymbol{U}_f \boldsymbol{h}_{t-1}) \\
\boldsymbol{i}_t &= \text{sigmoid}(\boldsymbol{W}_i \hat{\boldsymbol{x}}_t + \boldsymbol{b}_i + \boldsymbol{U}_i \boldsymbol{h}_{t-1}) \\
\boldsymbol{o}_t &= \text{sigmoid}(\boldsymbol{W}_o \hat{\boldsymbol{x}}_t + \boldsymbol{b}_o + \boldsymbol{U}_o \boldsymbol{h}_{t-1}) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tanh(\boldsymbol{W}_c \hat{\boldsymbol{x}}_t + \boldsymbol{b}_c + \boldsymbol{U}_c \boldsymbol{h}_{t-1}) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \circ \tanh(\boldsymbol{c}_t).
\end{aligned} \tag{3.10}$$

With $\hat{\boldsymbol{x}}_t \in \mathbb{R}^c$ being the input vector to the LSTM layer and $\boldsymbol{h}_t \in \mathbb{R}^m$ its output. $\boldsymbol{f}_t \in \mathbb{R}^m$, $\boldsymbol{i}_t \in \mathbb{R}^m$, and $\boldsymbol{o}_t \in \mathbb{R}^m$ denote the activation vectors for the forget gates, the input gates and the output gates respectively. The matrices $\boldsymbol{W} \in \mathbb{R}^{m \times c}$ are non-recurrent weight matrices. $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ are recurrent weight matrices for the previous hidden state $\boldsymbol{h}_{t-1} \in \mathbb{R}^m$ in the respective gates. The operator $\circ$ denotes the element-wise product. Note that the layer-wide $\boldsymbol{h}_{t-1}$ is an input for all individual cells, generating recurrent connections between them in an LSTM layer.

We use an element-wise

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{3.11}$$

as activation. Input and output squashing is done with the hyperbolic tangent. All weights $\boldsymbol{W}, \boldsymbol{b}$, and $\boldsymbol{U}$ are learnable.

It's also possible to stack various LSTM layers for more expressive power. This, however, didn't improve learning in our assembly scenarios, mainly due to the small data set sizes in comparison to that of very deep networks. We thus use a single-layered LSTM and adjust our skill model's complexity and parameter count with the number of cells m in that layer.

For an input sequence of consecutive points $\hat{\boldsymbol{x}}_{t-N}, \ldots, \hat{\boldsymbol{x}}_t$, Eq. (3.10) computes $N+1$ consecutive hidden states and cell states. Note that only for the last output has the LSTM seen all previous inputs, i.e. the full input sequence. We are only interested in these last states $\boldsymbol{h}_t, \boldsymbol{c}_t$ and define

$$\text{LSTM}(\hat{\boldsymbol{x}}_{t-N}, \ldots, \hat{\boldsymbol{x}}_t) := (\boldsymbol{h}_t, \boldsymbol{c}_t) \tag{3.12}$$

as a function to compute an encoding of a given input sequence.

The next section shows how this encoding is then used to model probability densities for the successor $\hat{\boldsymbol{y}}_t$.

3.3.2. Learning Probabilistic Behavior

The second component of our skill considers modeling humans' seemingly random trial and error. If we consider the LSTM model so far, and, adding a suitable output layer,

we could directly train supervised with minimizing $\sum_t (\hat{\boldsymbol{y}}_t - \boldsymbol{y}_t)^2$ for all t in our data set with $\boldsymbol{y}_t = \text{LSTM}(\hat{\boldsymbol{x}}_{t-n}, \ldots, \hat{\boldsymbol{x}}_t)$. The problem is that for continuous variables, such as our wrench vector $\hat{\boldsymbol{y}}_t$, a mean squared error function will implicitly learn the conditional average of the target data, conditioned on the input [189]. The skill model would, therefore, learn to imitate the average of the recorded strategies. As the data analysis showed with Fig. 3.9, this would equal a washing-out of the strategies' fine-grained characteristics. E.g. wiggling would become a mere pushing. To counteract this problem, we assume that in each time step, human strategies $\hat{\boldsymbol{y}}_t$ can be modeled as a probability density function (PDF) with a mixture of c-multivariate Gaussian distributions $\mathcal{N}_c(\boldsymbol{\mu}, \sigma^2)$.

Recapitulating some of the challenges from previous sections, our data conflict with the i.i.d assumption. In particular, the joint probability density $p(\hat{\boldsymbol{y}}_t, \hat{\boldsymbol{x}}_t)$ is time-dependent. Note that using a Gaussian Mixture Model (GMM) for regression, we could still approximate a static probability density function for our data to obtain the histogram plot Fig. 3.11. But it would fail with the sequences from Fig. 3.12 that require local coherence, once a random starting point is chosen. To assure local coherence during sampling from these distributions, the previous section introduced the LSTM-based encoding that shall provide the necessary semantic input for parameterizing our Gaussian mixture at each step. We use the architecture of Mixture Density Networks (MDN) [189] to connect the GMM with our LSTM layer.

Alternative approaches in literature without using a sequence encoder jointly encode spatial and temporal information directly into the GMM. The task's reproduction is then computed with Gaussian Mixture Regression (GMR) [102], see Section 2.2.5. While GMR would be possible in our approach from the parameterized GMM at each step, unbiased sampling from this distribution has advantages to overcome deadlocks. We discuss this in Section 3.3.5.

Another alternative is the combination of a Hidden Markov Model (HMM) with GMR, which we reviewed in Section 2.2.5. Being popular in literature, e.g. [104], [105], we briefly discuss differences to our approach: The idea is to use the HMM's temporal probabilistic evolution between state transitions in the computation of GMR. In this setting, each of the HMM's states covers a region in trajectory-space with a GMM, learned with an EM-based algorithm [101]. Specifically, the conventional GMR is adapted to use the HMM's *forward variable* [109] to include transition probabilities into the regression function [104], [105]. Using a first-order Markov model in these works, the incorporation of transition probabilities is limited to the last state only. Compared to our setting, this would be making only use of $p(\hat{\boldsymbol{y}}_t \mid \hat{\boldsymbol{x}}_t)$ in our notation, i.e. using a sequence encoder with length$= 1$.

Relying on the first-order Markov assumption is partly surprising, given that human performance generally seems correlated over longer sequences. In combination with a moderate number of states, this however quickly becomes computationally infeasible, see e.g. [190]. The trick is that the task's PDF is modeled to only comprise few states at specific landmarks during its execution. Then, observing single state transitions in the first-order Markov assumption covers a sufficient range in trajectory space. Note that this was mostly shown to work on comparatively unconstrained tasks with non-complex evolutions of distributions, such as table tennis strokes [104] or steering a rolling ball in a box through tilting [105].

In order to model assembly skills with high dynamics in force strategies with limited, in-contact motion, HMMs as temporal probabilistic models would require many states that parameterize many individual GMMs in order to resolve ambiguity and to model the

distribution complexity of Fig. 3.10. Whether this proves both computationally feasible and successful for low-clearance assembly with more complex insertion directions than peg-in-hole needs further research.

LSTM-based networks in contrast are not bound to a specific number of states, nor to a first-order Markov assumption to make computing a PDF for the dataset computationally feasible. Their recurrence can be interpreted as having a very high Markov order by default, covering hundreds of steps, and their MDN-part can parameterize a GMM at each prediction step anew. Being deep neural networks, however, they require more data for training but are almost unbounded in modeling the complexity of huge datasets. Both might be considered a drawback and an advantage at the same time. As a consequence of the advantage, the combination of LSTMs and MDNs has been used successfully in modeling and imitating highly complex human performance, such as authentic human handwriting from digital text [177] or full-body, freestyle dance motion [191].

Mixture Density Networks (MDN)

Introduced by Bishop [189], they model conditional probability densities of the target data, conditioned on the input data. Our notation describes this with $p(\hat{\boldsymbol{y}}_t \mid \boldsymbol{h}_t, \boldsymbol{c}_t)$, in which the hidden state $\boldsymbol{h}_t$ and cell state $\boldsymbol{c}_t$ are the encoding of the LSTM input sequence $\hat{\boldsymbol{x}}_{t-n}, n = N \dots 0$. To achieve this, MDNs are connected to neural network structures by feeding the network's last layer into the parameters of a mixture model that shall represent this probability density. In our case, we use the LSTM's encoding from Eq. (3.12) as input. For our scenario, the probability density should be interpreted as a density that, after integration over a given interval, yields how likely $\hat{\boldsymbol{y}}_t$ follows $\hat{\boldsymbol{x}}_{t-n}, n = N \dots 0$ in time across individual sequences of our datasets.

The conditional probability density becomes [189]

$$p(\hat{\boldsymbol{y}}_t \mid \boldsymbol{h}_t, \boldsymbol{c}_t) = \sum_{i=1}^{k} \alpha_i \phi_i(\hat{\boldsymbol{y}}_t \mid \boldsymbol{h}_t, \boldsymbol{c}_t) \ , \tag{3.13}$$

representing a linear combination of k kernel functions ϕ_i. We choose multivariate Gaussian distributions $\mathcal{N}_c(\boldsymbol{\mu}, \sigma^2)$ as c-variate kernel functions with $\alpha_i, \sigma_i \in \mathbb{R}$ and $\hat{\boldsymbol{y}}_t, \mu \in \mathbb{R}^c$, where $c = 6$ is the dimension of the final output feature vector of our network, such that the conditional density for the ith kernel becomes

$$\phi_i(\hat{\boldsymbol{y}}_t \mid \boldsymbol{\mu}_i, \sigma_i) = \frac{1}{(2\pi)^{c/2}\sigma_i^c} \exp\left(-\frac{\|\hat{\boldsymbol{y}}_t - \boldsymbol{\mu}_i\|^2}{2\sigma_i^2}\right). \tag{3.14}$$

Since the LSTM's output is used as MDN input, the final output parameter vectors $\boldsymbol{\alpha}, \boldsymbol{\sigma}, \boldsymbol{\mu}$ are in fact functions of the encoding $(\boldsymbol{h}, \boldsymbol{c})$, and thus functions of past sequences for each time step. During learning, this key feature makes it much easier for the probabilistic model to find suitable probability density functions that agree with our dataset and resolve ambiguity.

Fig. 3.14 illustrates the parametrization of the GMM. The parameters are composed of the MDN's output $\boldsymbol{z}$ that is fully connected to the LSTM's output with linear activation

$$\boldsymbol{z} = \boldsymbol{W}_z \left[\boldsymbol{h}^T, \boldsymbol{c}^T\right]^T . \tag{3.15}$$

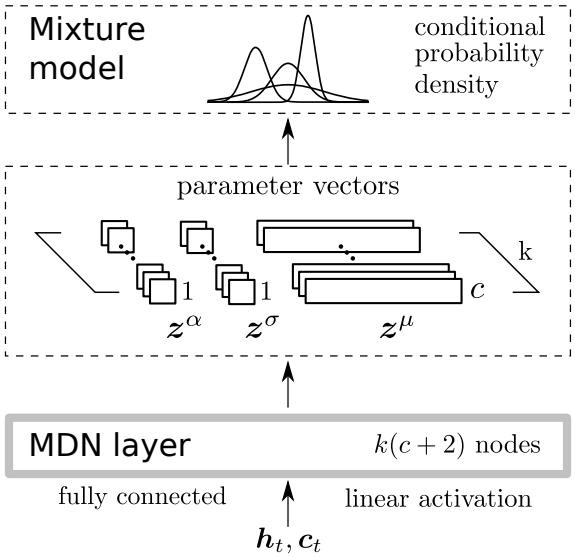


Figure 3.14.: Mixture Density Network (MDN) layer. The output z is split into individual vectors $z^\alpha, z^\sigma, z^\mu$, whose components parameterize the mixing coefficients, the standard deviations and means of a c-variate Gaussian mixture model with k basis functions.

$W_z \in \mathbb{R}^{k(c+2)\times 2m}$ is a learnable weight matrix of suitable dimension. For the parameterization of the Gaussians, z is split into a tuple $(z^\alpha, z^\mu, z^\sigma)$ and the individual components are used as follows, see [189]: The mixing coefficients sum up to 1 with a softmax

$$\alpha_i = \frac{\exp(z_i^\alpha)}{\sum_{j=1}^{k} \exp(z_j^\alpha)}\,, \qquad i = 1 \dots k \tag{3.16}$$

The centers of each kernel are

$$\mu_{ik} = z_{ij}^\mu\,, \qquad i = 1 \dots k\,, j = 1 \dots c \tag{3.17}$$

and the scale parameters are

$$\sigma_i = \exp(z_i^\sigma), \qquad i = 1 \dots k \tag{3.18}$$

In principle, it's possible to use full covariance matrices $\mathbf{\Sigma} \in \mathbb{R}^{c\times c}$ for each of the c-variate Gaussians $\mathcal{N}_c$ instead of the scalar σ_i. Following Bishop's considerations [189] and our own experimentally analysis of learning performance on the toy dataset, we use the simpler and computationally faster version in our skill model.

Choosing the number k of Gaussian kernels is difficult to derive analytically. Bishop states that using many kernels does not harm, since the network can switch off redundant ones by parameterizing low mixing coefficients or overlap kernels by choosing similar μ and σ [189]. In practice, a conservatively high number of kernels might however be bound by computation performance requirements, such that a trade-off has to be made. We achieved good performance for $k = 4, 8$ in our experiments and recommend those as default also for more complex assembly tasks.

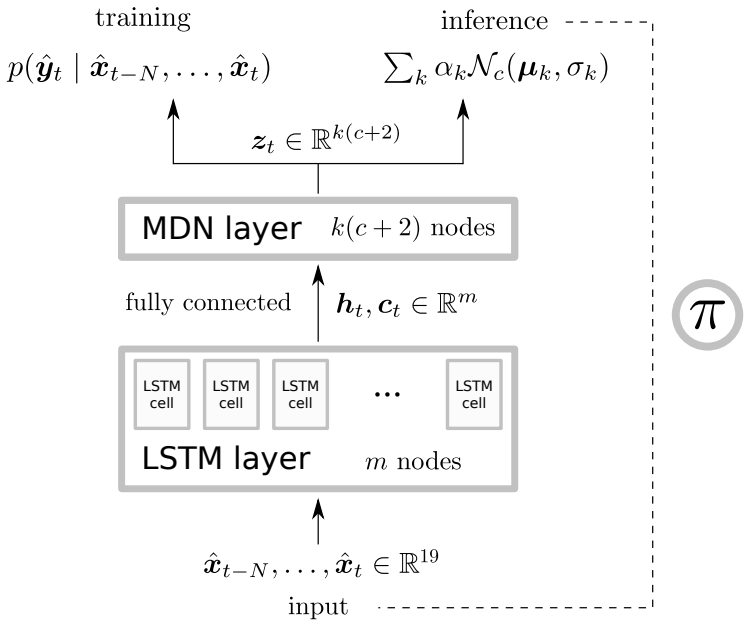


Figure 3.15.: Architecture for modeling assembly skills π. This many-to-one mapping is learned supervised from human behavior, recorded as sequences of state-action pairs. The model has two modes, indicated by the two output branches: Training and inference.

3.3.3. Skills as Probabilistic Forward Models

Fig. 3.15 shows the final skill model as the combination of both components: An LSTM-network for representing past behavior as a compact sequence encoding and an MDN-network for modeling probability densities of next strategies. Seen on a higher level, this architecture shall enable to learn human behavior from the recorded datasets and then imitate its core strategies for challenging assembly skills. Our concept based human skills on mental forward models in physiological motor control. In analogy, we propose to think of the technical realization of assembly skills as *probabilistic forward models*, that we implement as recursive prediction models from state-action pairs to strategies. Their output forms promising motor control commands $\boldsymbol{f}^h$ that cause part motion $\boldsymbol{x}, \dot{\boldsymbol{x}}$, which is then perceived and reacted upon.

On the implementation side, the skill model from Fig. 3.15 has two output modes: The left branch describes the training, which makes use of the conditional probability density and is described in Section 3.3.4. And inference, which uses the mixture model's parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\alpha}$ to obtain concrete wrench vectors as predictions. Several methods exist for regression and Section 3.3.5 discusses the implications of two important ones: GMR as a least-squares estimate of the Gaussians' centers that is frequently used in LfD, and stochastic sampling that introduces certain randomness at each step.

Training Data Composition

Our concept proposed the recording of human behavior during assembly into individual demonstrations $\{\xi_0, \xi_1, \ldots\}$. These demonstrations are discretized sequences of individual data points, each represented by the tuple of features $[\boldsymbol{x}_t, \dot{\boldsymbol{x}}_t, \boldsymbol{f}_t^h]$. We used the discrete

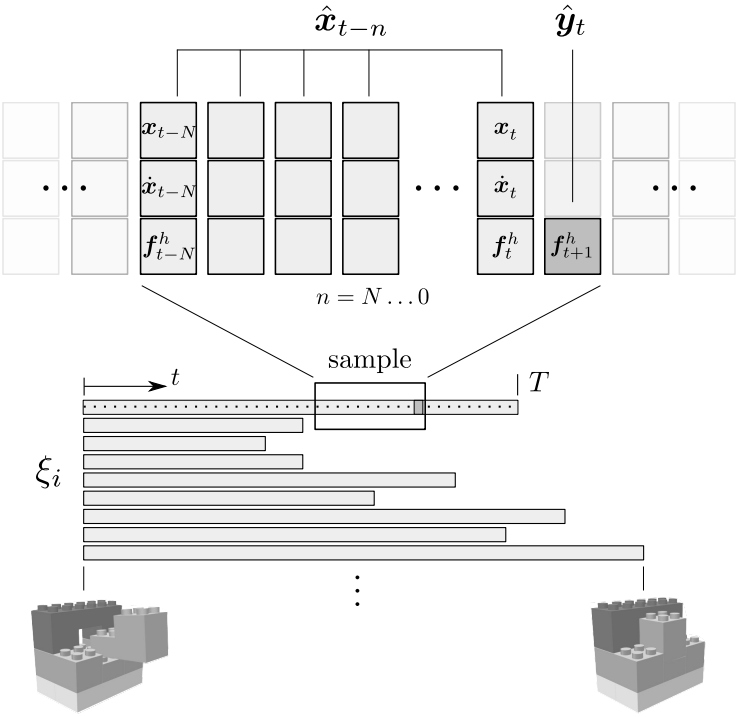


Figure 3.16.: Training data composition. We randomly extract samples as discretized subsequences out of complete assembly demonstrations ξ. Each data point comprises the tuple $\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{f}^h$. In each sample, the last data element $\hat{\boldsymbol{x}}_{t+1}$ holds the label $\hat{\boldsymbol{y}}_t$ (highlighted) for the previous sequence $\hat{\boldsymbol{x}}_{t-n}$.

index t to refer to a specific instance, with $t \in [0 \dots T]$ and T being the number of data points in that demonstration. T differs for each demonstration ξ mainly due to varying human performance for the task and partially due to using random starting poses of the active assembly objects. This shows that these recordings are in fact not optimal and reflect our concept of learning *human* behavior, rather than learning optimal behavior. We investigate the impact of the quality of demonstrations in Section 5.1.6, in which we cluster the demonstrations for their duration and train our assembly skill network on individual clusters for comparison.

Before composing actual training data out of demonstrations, we perform standardization on the whole dataset for each feature's dimension separately. This assures the features to have a zero mean and a standard deviation of 1 while being all at the same scale to facilitate learning [192]. We obtain *samples* from the data set by randomly selecting ξ out of the entirety of assembly demonstrations and t as a random pivot in that demonstration. The sample is then composed of the input sequence $\hat{\boldsymbol{x}}_{t-N}, \dots, \hat{\boldsymbol{x}}_t$ and the according label $\hat{\boldsymbol{y}}_t$, which is the wrench component of the next $\hat{\boldsymbol{x}}_{t+1}$. Fig. 3.16 illustrates this process. This produces randomly extracted, uncorrelated samples of equal length $N + 2$, where N is the number of steps that the LSTM-MDN network can use to predict the corresponding probability density function for the label. During training, this sampling process will be used to form *batches* of samples for parallel processing. Note that the samples may partially overlap in these batches and cover the entirety of the training data stochastically. Our assumption behind this approach is that assembly skills manifest

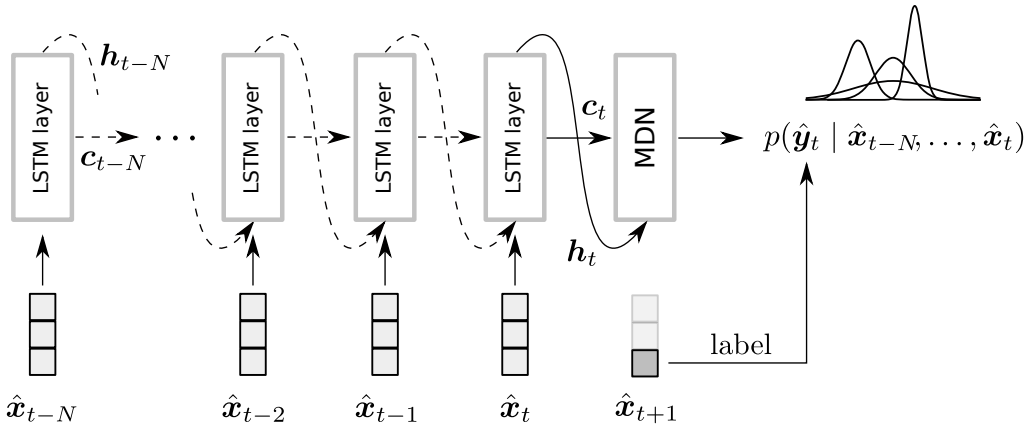


Figure 3.17.: Unrolling the skill model for training with Backpropagation Through Time (BPTT). Arrows show the LSTM layer's input at each time step and the dashed arrows indicate the formerly recurrent connections that have become normal feedforward connections.

themselves already in shorter subsequences (strategies) and do not necessarily require complete demonstrations to learn human behavior. We investigate N as a hyperparameter for learning performance in Section 5.1.4. In combination with the recording rate in simulation, we also obtain an estimate of how long a strategy takes on average during assembly.

The next sections describe the training mechanism in the context of modern machine learning libraries and how skills make predictions during inference.

3.3.4. Training

For training neural networks, the early backpropagation (BP) algorithm [193] is still widely used. The algorithm has been re-invented in various domains, see e.g. the work of Werbos for a discussion [194]. It is a gradient-based, iteratively working optimization algorithm with four principal steps [192]:

(i) The *forward pass* computes the network's prediction for a given input
(ii) The *loss function* computes an error by measuring the output against a label
(iii) The *reverse pass* computes each variable's error contribution as a gradient, using partial derivatives
(iv) The *gradient descent* step corrects the learnable weights with a small increment using the gradient

Recurrent neural networks require an additional step to make them accessible for BP. The combination is referred to as Backpropagation through time (BPTT): During training, the skill network is *unrolled* for N time steps and becomes a deep feedforward network as illustrated in Fig. 3.17. For our skill model, the LSTM layer's recurrent connections from Fig. 3.13 become normal feedforward connections and contribute along with the layer's input at each step. The various instances of the LSTM layer from Fig. 3.17 all refer to the same layer, illustrated at various time steps, with only one set of learnable weights from Eq. (3.10).

Forward Pass

For the input sequence $\hat{\boldsymbol{x}}_{t-N}, \ldots, \hat{\boldsymbol{x}}_t$ from one given sample, the network predicts the conditional probability density $p(\hat{\boldsymbol{y}}_t \mid \hat{\boldsymbol{x}}_{t-N}, \ldots, \hat{\boldsymbol{x}}_t)$. It does this by using the LSTM-based encoder with Eq. (3.12) to obtain the sequence encoding $\boldsymbol{h}_t, \boldsymbol{c}_t$, followed by the MDN equations Eq. (3.15) to Eq. (3.18) to obtain the output $[\boldsymbol{z}^\alpha, \boldsymbol{z}^\sigma, \boldsymbol{z}^\mu]$ for parameterization of the mixture model. And finally by using Eq. (3.14) and Eq. (3.13) to compute the conditional probability density for the given label $\hat{\boldsymbol{y}}_t$. In this complete pass through the neural network, all nodes' values are stored in a bookkeeping process for the summation of gradients in the reverse pass.

Loss Function

The loss function $\mathcal{L}$ is a negative log-likelihood, using the conditional probability densities p, conditioned on the sample's input sequence and evaluated for the sample's label:

$$\mathcal{L}(\boldsymbol{\theta}, \hat{\boldsymbol{x}}_{t-N}, \ldots, \hat{\boldsymbol{x}}_t, \hat{\boldsymbol{y}}_t) = -\ln\left(p(\hat{\boldsymbol{y}}_t \mid \hat{\boldsymbol{x}}_{t-N}, \ldots, \hat{\boldsymbol{x}}_t)\right) \rightarrow \boldsymbol{E}(\boldsymbol{\theta}). \tag{3.19}$$

The result is an error $\boldsymbol{E}$, that also depends on the current choice of learnable weights, which we denoted with the combined vector $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]^T$. These learnable weights compose the weight matrices for the node connections of our skill network from Eq. (3.10) and Eq. (3.15).

This loss offers an interpretation of the network's ability to parametrize a suitable density around the given label, i.e. around the ground truth. After integration over a given interval, this density would yield the probability of observing the label. A high loss indicates that the network estimated a low density around the label with a high variance of the Gaussians. Conversely, a low loss indicates that the network can accurately estimate the label's distribution for that sequence, and does this with high confidence. Put in simple terms, using BP to train the network with $\mathcal{L}$ can be interpreted as forcing the network to learn to find the labels and thus the recorded sequences of strategies likely by predicting distributions that best explain the recorded dataset.

Reverse Pass

After having computed the error, the next step is to compute each parameter's contribution to that error with partial derivatives, i.e. $\partial \boldsymbol{E} / \partial \theta_i$. In modern machine learning libraries, computing these derivatives works iteratively, relying on Automatic Differentiation (AD), a scheme that uses the chain rule and is a powerful alternative to numeric differentiation in computer programs [195]. This is applicable for programming languages in general and is leveraged as reverse mode AD in Tensorflow's C++ core [188], [192], which we use in this thesis.

For each node's operation in the network, this scheme needs to know the partial derivatives w.r.t to the node's predecessors. They are given in [189] for the MDN part and in [175] for the LSTM part. They must be implemented into a computational graph of basic functions. AD then computes $\partial \boldsymbol{E} / \partial \theta_i$ reversely by concatenation through the complete computational graph from the network's output to the input, evaluating the partials for the current choice of parameters $\boldsymbol{\theta}$, and each node's value, determined for the current sample $(\hat{\boldsymbol{x}}_{t-n}, \hat{\boldsymbol{y}}_t)$ in the forward pass.

The combination of the evaluated partial derivatives is the error gradient that will be used in the gradient descent step for the actual weight update:

$$\nabla \boldsymbol{E}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial \boldsymbol{E}(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial \boldsymbol{E}(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots \\ \frac{\partial \boldsymbol{E}(\boldsymbol{\theta})}{\partial \theta_n}. \end{bmatrix} \tag{3.20}$$

We use Mini-Batch Gradient Decent in our supervised learning setting, i.e. we compute the gradient for a batch of q training samples $(\hat{\boldsymbol{x}}_{t-n}, \hat{\boldsymbol{y}}_t)^q$ and according weights $\boldsymbol{\theta}^q$ jointly with

$$\nabla \boldsymbol{E}^q(\boldsymbol{\theta}^q) = \sum_{i=1}^{q} \nabla \boldsymbol{E}(\boldsymbol{\theta}_i) \tag{3.21}$$

for each computation step. The advantage is a trade-off between speed and robustness of the two corner cases Stochastic Gradient Descent and Batch Gradient Descent, see e.g. [192]. Qualitative influence of the batch size on learning performance and generalization is discussed e.g. in [196], [197], of which the latter proposes sizes in the range of 2 to 32 as good defaults. We form the batch of samples in each training step online by obtaining and gathering samples as explained in Section 3.3.3.

Gradient Descent Step

Having the gradients as the joint vector of partial derivatives for all variables, the final step is to incrementally upgrade them with

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla \boldsymbol{E}^q(\boldsymbol{\theta}^q). \tag{3.22}$$

An optimizer handles setting the learning rate η and other learning hyperparameters (as function arguments for η). We use Adam [198] and its implementation in Tensorflow. Learning performance is discussed in Section 5.1.

3.3.5. Inference

In this thesis' approach, the assembly skill is now fully trained and there is no further online learning. During inference, the skill model predicts a parameterization for a GMM that can mimic human strategies. Gaussian mixture regression typically covers several methods to obtain single estimates from the density, such as least-squares estimates (LSE) or stochastic sampling [102]. Both are established methods and need a brief discussion for the tasks considered.

Regression with Least-Squares Estimates

LSE is often used as regression for LfD tasks, e.g. [100], [104], [105]. In this method, an individual estimate is computed from the Gaussian mixture by computing an average of conditional expectations. This makes LSE deterministic during predictions and returns

the mean of alternatives for inverse mappings [102]. As elaborated in Section 3.3.2, a least-squares approximation for regression will, therefore, average and lose important strategies for our assembly tasks.

Regression with Sampling

In contrast to an LSE, sampling returns random draws from the full distribution. This randomness can also prove effective against deadlocks for peg-in-hole assembly [108]. Note that conventional GMMs are only suitable in this setting for sampling if the task is simple enough to tolerate fluctuating control commands. The broad coverage of trajectory space with few Gaussians makes it hard to achieve consistent sequences across samples. In contrast, through the more expressive power of the LSTM-encoder, our model can build fine-grained distributions along the trajectories *on the fly* and provide coherence despite sampling also for complex insertions.

Sampling from our skill model output can be implemented in a two-step approach:

(i) First, the MDN's mixture coefficients α_i form a categorical distribution. Through building the cumulative distribution function (CDF) and using uniform sampling from $\mathcal{U}(0,1)$, the closest number in the CDF to the draw selects an individual Gaussian $\mathcal{N}_c$ of the mixture model.

(ii) In the second step, the Box-Muller algorithm [199] is applied for each of the c-variate Gaussian's dimensions to obtain $\boldsymbol{f}^h \in \mathbb{R}^c$. This algorithm turns a random auxiliary sample a from a *uniform* distribution $a \sim \mathcal{U}(-1,1)$ into a sample b from a *normal* distribution $b \sim \mathcal{N}(0,1)$. Re-scaling with $b\sigma + \mu$ reflects the MDN's parameterization for that dimension.

We use the Tensorflow distribution library [200] for this task.

3.4. Assembly Skills

The formulation of assembly skills as probabilistic forward models was robot-independent. For practical applications of these skills, the robot takes the simulator's role, i.e. turning $\boldsymbol{f}^h$ into motion and reporting that motion as part-relative feedback. Fig. 3.18 shows the according control scheme with inputs and outputs. Note that feedback for the skill does not contain measured force-torque signals from the robot, which was a design decision in our concept of seeking to implement human-inspired, mental forward models. We also proposed this as a means to bridge the reality gap by bypassing the simulator's flaws in contact force rendering. In each step, we obtain a prediction through sampling that needs transformation into adequate coordinate systems before being processed as reference setpoints in the robot controller. This is handled in Section 3.4.1. Section 3.4.2 explains how strategies, i.e. force-torque control commands can be used on top of conventional, compliant trajectory execution.

3.4.1. Assembly Task Specification

Previous sections used formulations in task space, which described the pose $\boldsymbol{x}$ and velocity $\dot{\boldsymbol{x}}$ of the active assembly object in frame $\mathcal{O}$ of the passive assembly object. This task-relative consideration was robot manipulator-independent and supports applying

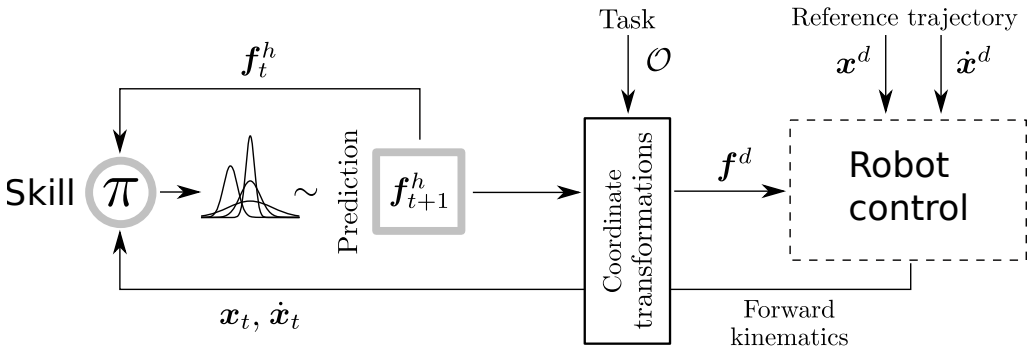


Figure 3.18.: Skills as closed-loop controllers for strategies during the assembly process. The force-torque setpoints neatly integrate with reference trajectories from other sources, such as DMPs, motion planning, or point-to-point teaching.

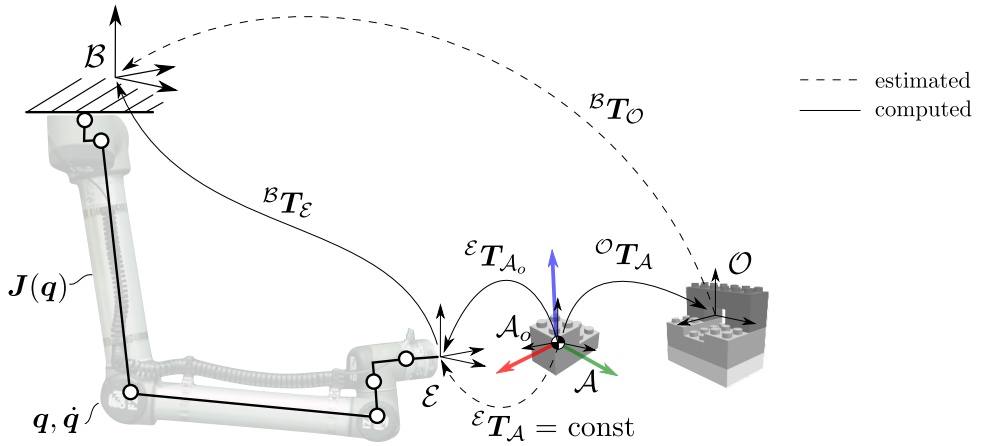


Figure 3.19.: Coordinate frames and transformations for an assembly task. The transformations relate the robot's base $\mathcal{B}$ with the assembly's origin $\mathcal{O}$, the active assembly part's local frames $\mathcal{A}, \mathcal{A}_o$, and the robot's end-effector frame $\mathcal{E}$.

skills disjoint from the robot's configuration space, providing an alternative approach to the correspondence problem. For deploying a skill on robotic systems, further coordinate frames and transformations are required to display the skill's predictions in a reference frame suitable for robot controllers, and in turn displaying the robot's feedback in a frame where the neural network skill model expects its inputs.

We use homogeneous transformations of the scheme ${}^{\text{to}}\boldsymbol{T}_{\text{from}}$ to describe transformations between individual frames. In this scheme, transformations are composed with

$$ {}^{\text{to}}\boldsymbol{T}_{\text{from}} = \begin{bmatrix} {}^{\text{to}}\boldsymbol{R}_{\text{from}} & {}^{\text{to}}\boldsymbol{r} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}, \tag{3.23} $$

using the rotation matrix ${}^{\text{to}}\boldsymbol{R}_{\text{from}}$ between the two frames and the Cartesian vector between the two frame origins ${}^{\text{to}}\boldsymbol{r}$, displayed in the target frame. We will extract both components from their respective homogeneous representation when transforming the spatial quantities $\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{f}^h$. The inverses of transformations are computed with

$$ ({}^{\text{to}}\boldsymbol{T}_{\text{from}})^{-1} = {}^{\text{from}}\boldsymbol{T}_{\text{to}} = \begin{bmatrix} {}^{\text{to}}\boldsymbol{R}_{\text{from}}^T & -{}^{\text{to}}\boldsymbol{R}_{\text{from}}^T\,{}^{\text{to}}\boldsymbol{r} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}, \tag{3.24} $$

Fig. 3.19 shows important frames and coordinate transformations with the following meaning:

${}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{O}}$: Describes the passive assembly object's origin with respect to the robot base. In an autonomous setting, the robot might acquire this transformation e.g. with visual sensors, including uncertainty. For most use cases, this transformation can be considered constant throughout the assembly. The concept of using origin-centered skills, however, also allows for a moving origin, e.g. on conveyor belts.

${}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{E}}$: This transformation relates the base with the end-effector. The transformation depends on the joint angle vector $\boldsymbol{q}$ and is continuously computed in robot control with forward kinematics.

${}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{A}}$: This constant transformation relates the active object's pose with respect to the robot's end-effector. It is assumed constant throughout the assembly task and initially given during setup. In practical use cases, robotic grippers will hold the active objects during assembly, such that these grasps are estimated, albeit usually with higher accuracy than ${}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{O}}$.

${}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{A}_o}$: This transformation is dynamically built from the active object's origin with respect to the robot's end-effector and the passive object's rotation, given in end-effector coordinates.

${}^{\mathcal{O}}\boldsymbol{T}_{\mathcal{A}}$: The relative pose of the active assembly object w.r.t to the passive target. This dynamically changing transformation is semantically equivalent to the pose $\boldsymbol{x}$.

We use the following convention to specify our quantities: Without a leading superscript, the quantities are assumed to be given within skill-centric frames, i.e. $\boldsymbol{x} = {}^{\mathcal{O}}\boldsymbol{x}, \dot{\boldsymbol{x}} = {}^{\mathcal{O}}\dot{\boldsymbol{x}}$ and $\boldsymbol{f}^h = {}^{\mathcal{A}_o}\boldsymbol{f}^h$. This applies to data recorded in the simulator and to the inputs and outputs of the skill model π. In contrast, a trailing superscript ^{d} indicates the desired target value as a setpoint for robot control in a suitable frame, specifically $\boldsymbol{x}^d = {}^{\mathcal{B}}\boldsymbol{x}, \dot{\boldsymbol{x}}^d = {}^{\mathcal{B}}\dot{\boldsymbol{x}}$ in the robot's base frame $\mathcal{B}$ and $\boldsymbol{f}^d = {}^{\mathcal{E}}\boldsymbol{f}^h$ in its end-effector frame $\mathcal{E}$.

Robot Control from Skill Outputs

Using the transformation matrices from Fig. 3.19 and its components, users obtain target setpoints for robot control from skill predictions as follows: The target wrench for force tracking is the equivalent wrench of $\boldsymbol{f}^h$, transformed to the robot's end-effector. We use spatial transformations according to Murray et al [41] with

$$\boldsymbol{f}^d = {}^{\mathcal{E}}\boldsymbol{f}^h = \begin{bmatrix} {}^{\mathcal{E}}\boldsymbol{R}_{\mathcal{A}_o} & \boldsymbol{0} \\ -{}^{\mathcal{E}}\boldsymbol{R}_{\mathcal{A}_o}{}^{\mathcal{E}}\widehat{\boldsymbol{r}} & {}^{\mathcal{E}}\boldsymbol{R}_{\mathcal{A}_o} \end{bmatrix} \boldsymbol{f}^h \,, \tag{3.25}$$

in which ${}^{\mathcal{E}}\boldsymbol{R}_{\mathcal{A}_o} = {}^{\mathcal{E}}\boldsymbol{R}_{\mathcal{O}} = {}^{\mathcal{E}}\boldsymbol{R}_{\mathcal{B}}{}^{\mathcal{B}}\boldsymbol{R}_{\mathcal{O}}$. The components of ${}^{\mathcal{E}}\widehat{\boldsymbol{r}}$ are extracted from ${}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{A}}$, and the hat operator $\widehat{\ }$ is used to represent the cross product as matrix multiplication [41]:

$$\widehat{\boldsymbol{r}} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} . \tag{3.26}$$

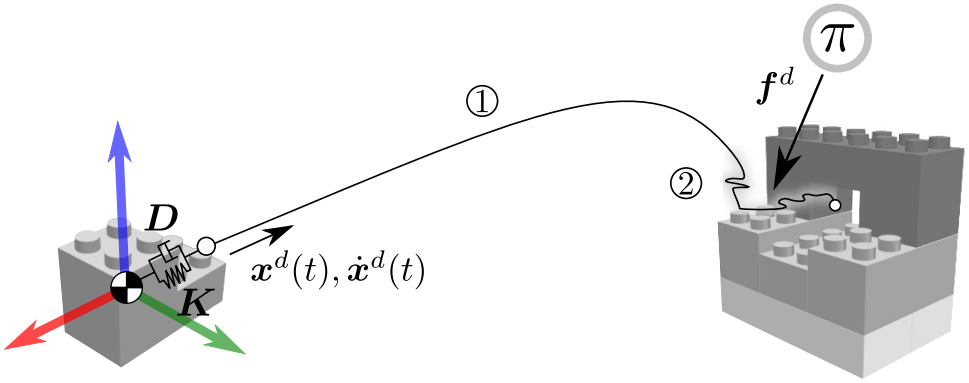


Figure 3.20.: Compliant trajectory execution with strategy overlay. Assembly tasks are divided into two principal phases: ① the execution of nominal target trajectories with end-effector impedance and ② wrench-based strategy overlay to dynamically adapt to unforeseen jamming and wedging.

Skill Inputs from Robot Control

For obtaining skill inputs from robot control we have:

$$\boldsymbol{x} = {}^{\mathcal{O}}\boldsymbol{x} \leftarrow {}^{\mathcal{O}}\boldsymbol{T}_{\mathcal{A}} = {}^{\mathcal{O}}\boldsymbol{T}_{\mathcal{B}}{}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{E}}{}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{A}}, \tag{3.27}$$

in which ${}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{E}}$ is computed by a forward kinematics routine. And

$$\dot{\boldsymbol{x}} = {}^{\mathcal{O}}\dot{\boldsymbol{x}} = \begin{bmatrix} {}^{\mathcal{O}}\boldsymbol{R}_{\mathcal{B}} & {}^{\mathcal{O}}\widehat{\boldsymbol{r}}\,{}^{\mathcal{O}}\boldsymbol{R}_{\mathcal{B}} \\ \boldsymbol{0} & {}^{\mathcal{O}}\boldsymbol{R}_{\mathcal{B}} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\omega}}\,{}^{\mathcal{B}}\boldsymbol{r} + \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} = \boldsymbol{J}\dot{\boldsymbol{q}}. \tag{3.28}$$

$\boldsymbol{J}(\boldsymbol{q})$ is the manipulator's Jacobian for the end-effector. ${}^{\mathcal{O}}\boldsymbol{R}_{\mathcal{B}}$ and ${}^{\mathcal{O}}\boldsymbol{r}$ are taken from ${}^{\mathcal{O}}\boldsymbol{T}_{\mathcal{B}} = ({}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{O}})^{-1}$. The vector ${}^{\mathcal{B}}\boldsymbol{r}$ is taken from ${}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{E}}{}^{\mathcal{E}}\boldsymbol{T}_{\mathcal{A}}$ as the origin of $\mathcal{A}$ given in the robot's base. Note that $\boldsymbol{f}^h$ is recurrently connected between skill output and input and needs no transformation.

3.4.2. Trajectories with Strategy Overlay

Assembly skills are well suited to be integrated on top of other commonly used methods. One condition is that the application's robot control can make use of explicit target wrenches $\boldsymbol{f}^d$. As investigated in Section 2.1, suitable schemes could be based both on hybrid and parallel force/motion control and admittance control. We consider the last for illustration purposes.

When applying assembly skills on the robotic platform, we distinguish two principal control phases: Phase ① a free space motion towards the assembly's origin of interest, and phase ② the actual insertion after establishing the first contact. Fig. 3.20 illustrates both phases. In ①, the robot executes its nominal trajectory $\boldsymbol{x}^d(t), \dot{\boldsymbol{x}}^d(t)$ without skill intervention. This thesis' approach is relatively independent of where these trajectories come from. Possible options include point-to-point teaching in a calibrated setting, motion planning in a simulation environment, or generalization with DMPs. In phase ②, the skill's prediction partially overrule restoring forces as described at the beginning of the chapter's concept. The computed net force $\boldsymbol{f}$ from Eq. (3.1) can then be transformed into

the end-effector frame with Eq. (3.25) and regulated in a suitable force control law. Applying the human-inspired strategies for this task alters the nominal trajectory in contact, as illustrated in Fig. 3.20. The learned skill reacts dynamically, observing both the assembly's progress and its previous actions to imitate human behavior. Switching from phase ① to phase ② and enabling the associated assembly skill can be delayed until e.g. reaction forces exceed defined safety thresholds. Exceeding these limits would indicate that the robot encountered unexpected resistance during execution and diverges significantly from its nominal target trajectory, so that recovering without strategies seems unlikely.

Applicability

Using the skill's predicted target wrenches is widely applicable to existing control schemes, such as Hybrid Force/Position control from Section 2.1.2 and Admittance Control from Section 2.1.3. It is also natively applicable to torque-based control schemes of lightweight robots and thus makes a contribution beyond this thesis' focus on motion-actuated systems.

3.5. Summary and Conclusion

So far, this thesis has proposed an intuitive method to model object-centric assembly skills, whose predictions can solve part jamming/wedging on compliant controllers. On a higher level of abstraction, this thesis followed the idea of modeling and capturing skills as human-inspired, mental forward models. The skills shall learn implicit strategies within recorded datasets for tight-fitting assemblies. The usage of simulation supports offline programming in an industrial context but required trade-offs between realism and numerical stability. The goal was to obtain simulation environments, in which operators can naturally steer the objects but are forced to apply strategies under artificially increased friction. This shall bridge the reality gap and enable transfer to real-world scenarios. Based on data analysis, we proposed a recurrent, probabilistic neural network architecture for modeling these assembly skills. The skills' formulation is robot-independent and object-centric in task space, and the transformation into robot-related coordinate frames can be embedded into automatic routines. Robot controllers can use the predicted strategies in form of setpoints and overlay existing methods for compliant trajectory execution with these force profiles. The next chapter continues the skill's idea of forward models and simulation and contributes a new control paradigm for skill execution to the field of admittance control.

4. Compliant Skill Controllers

Assembly skills so far learned from data, generated in simulation environments. They are robot-independent and exist in task space around two assembly objects as sequential-probabilistic force-torque strategies. Deploying them now requires controllers that execute these skills on motion-actuated robots. A step that can be interpreted as solving the correspondence problem from Section 3.2.3 on the control level.

This chapter describes that way from human-inspired skills to robotic skill controllers on motion-actuated robots. We propose a new mapping paradigm for these systems and contribute a new compliant controller for assembly skills and beyond: Section 4.1 introduces the overall concept that fundamentally bases on forward dynamics simulations on a virtual model, which is the core of Section 4.2. Designing and implementing a new compliant controller under this rationale is derived in Section 4.3. Section 4.4 closes the big picture with further discussions.

4.1. Concept

The previous Chapter 3 on human-inspired skills made implicit assumptions on robot control while learning from simulation-based data: The active assembly object was teleoperated along perfectly decoupled control axes in task space under velocity-proportional damping. Our approach was to learn the skills in an object-centric and robot-independent way by delegating control to the simulation's physics engine. This separation of skill and robot control was an important part of our concept for tackling the correspondence problem in PbD in robotics. Executing the skills on a robot now requires a similar performance: Robot control must overtake the physics engine's role with imitating rigid-body motion due to teleoperated force-torque commands, and measured contact forces at the robot's wrist replace the contact simulation of the physics engine. In addition, possible motion setpoints from trajectory execution must be included.

We briefly reconsider control schemes from Chapter 2 under these requirements and then propose a new approach, specifically tailored for our skill formulation on motion-actuated systems.

4.1.1. Options for Robot Control

Section 3.4.1 formulated the skills' predictions that become setpoints for robot control. Fig. 4.1 illustrates an example manipulator with relevant quantities. Current control schemes can partially include these setpoints: Hybrid Force/Position control from Section 2.1.2 natively supports reference velocities and target wrenches. However, the need for precisely describing time-varying selection matrices is a drawback under uncertainty and makes this option less flexible. Parallel Force/Position control circumvents the need for orthogonality, but like Hybrid Force/Position control does not support desired reference poses $\boldsymbol{x}^d$. Both use integral and proportional-integral gains for velocity and force

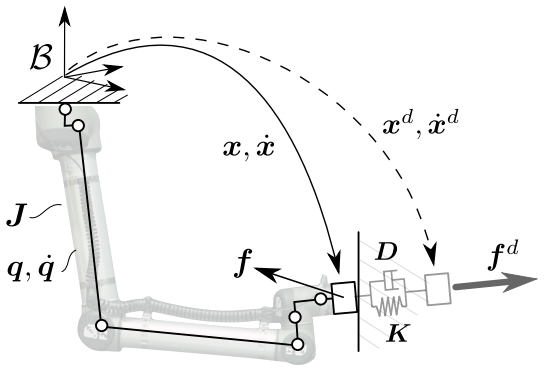


Figure 4.1.: Motion-actuated robot in rigid contact. A virtual spring-damper element puts current state $\boldsymbol{x}, \dot{\boldsymbol{x}}$ and desired state $\boldsymbol{x}^d, \dot{\boldsymbol{x}}^d$ into relation with force-torque sensor measurements $\boldsymbol{f}$ and target wrench $\boldsymbol{f}^d$.

tracking, respectively. Admittance control from Section 2.1.3 supports both reference poses and velocity. Numerical integration of a first-order system in Cartesian coordinates with the excitation $\boldsymbol{f}$ yields the reference velocity $\dot{\boldsymbol{x}}^r$. Like the simpler schemes of Compliance and Damping control, force control is achieved implicitly and $\boldsymbol{f}^d$ is not directly included as the desired reference. However, adaptations to these schemes could use $\Delta\boldsymbol{f} = \boldsymbol{f}^d - \boldsymbol{f}$ instead and provide this option. Table 4.1 shows an overview of the schemes.

Table 4.1.: Control approaches and inclusion of desired reference setpoints.

Scheme	control law	$\boldsymbol{x}^d$	$\dot{\boldsymbol{x}}^d$	$\boldsymbol{f}^d$
Hybrid Force/Position control Eq. (2.4)	$\dot{\boldsymbol{x}}^r = \boldsymbol{S}_x\dot{\boldsymbol{x}}^c + \boldsymbol{D}^{-1}\boldsymbol{S}_f\boldsymbol{f}^c$		✓	✓
Parallel Force/Position control Eq. (2.5)	$\dot{\boldsymbol{x}}^r = \dot{\boldsymbol{x}}^c + \boldsymbol{D}^{-1}\boldsymbol{f}^c$		✓	✓
Admittance control Eq. (2.6)	$\dot{\boldsymbol{x}}^r \xleftarrow{\text{int.}} \boldsymbol{D}\Delta\dot{\boldsymbol{x}} + \boldsymbol{K}\Delta\boldsymbol{x} = \boldsymbol{f}$	✓	✓	(✓)
Compliance control Eq. (2.7)	$\Delta\boldsymbol{x}^r = \Delta\boldsymbol{x} - \boldsymbol{K}^{-1}\boldsymbol{f}$	✓		(✓)
Damping control Eq. (2.8)	$\dot{\boldsymbol{x}}^r = \dot{\boldsymbol{x}}^d - \boldsymbol{D}^{-1}\boldsymbol{f}$		✓	(✓)

With slight modifications, all schemes support using the target wrenches $\boldsymbol{f}^d$ and make the assembly strategies widely applicable as an overlay to existing trajectory execution. Admittance control provides the most flexibility in making use of all setpoints. As a drawback, however, all approaches require stable alternatives to the plain Jacobian inverse to assure stability near singular configurations. This is due to first computing $\dot{\boldsymbol{x}}^r$ in Cartesian space and then depending on the robot's ability to render this reference velocity through joint actuation. A strong alternative that has however been largely unconsidered, is extending the numerical integration of the Admittance control scheme to robot dynamics and circumventing the inherent stability problem of Inverse Kinematics.

4.1.2. A New Mapping Paradigm

Similar to impedance control, we substitute the target setpoints $\boldsymbol{x}^d, \dot{\boldsymbol{x}}^d, \boldsymbol{f}^d$, the current end-effector state $\boldsymbol{x}, \dot{\boldsymbol{x}}$, and measured contact wrench $\boldsymbol{f}$ into a first order system to obtain

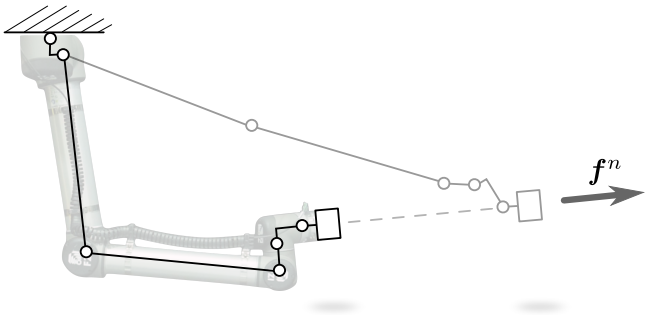


Figure 4.2.: Schematic illustration of forward dynamics simulation. The net force $\boldsymbol{f}^n$ acts as external load on the end-effector, and the mechanism stays stable even in singular configurations.

an overall net force $\boldsymbol{f}^n$ with

$$\boldsymbol{f}^n := \boldsymbol{f}^d - \boldsymbol{f} + \boldsymbol{K}(\boldsymbol{x}^d - \boldsymbol{x}) + \boldsymbol{D}(\dot{\boldsymbol{x}}^d - \dot{\boldsymbol{x}}) \,. \tag{4.1}$$

Eq. (4.1) holds for both in-contact and free motion. The central idea of our approach is to simulate the manipulator as an articulated, rigid body system in zero gravity, apply $\boldsymbol{f}^n$ at the end-effector, and compute the robot's reaction motion. The result serves as the control signal for the real actuators. Different from numerical integration and IK solving in admittance control, this method leads to inherently stable mappings from Cartesian to joint space. Fig. 4.2 illustrates this effect for a special case: Back-drivable gears allow the simulated manipulator to yield the net force $\boldsymbol{f}^n$ as good as possible. To deepen this thought experiment, we assume the links to be sufficiently lightweight. Constraints of the kinematic chain naturally limit this motion and the robot's articulations mechanically compensate external loads in singularity.

The next step is to mimic the behavior of Fig. 4.2 with forward dynamics algorithms and design an adequate model for motion-actuated systems.

4.2. Virtual Forward Dynamics Models

Unlike inverse dynamics algorithms, which are established for torque control on robotic systems, *forward dynamics* algorithms have more application in rigid body simulation and are less explored in the literature for manipulator control. The following sections show how they are particularly suitable for motion-actuated systems in form of direct mappings from wrench space to joint accelerations: Section 4.2.1 first introduces basic principles and algorithmic options. Section 4.2.2 then derives simplifications that fit them to our requirements of robot control. A feature of motion-actuated systems is their compensation of load dynamics in the joints, so that outer control loops are not required to compensate for these terms. In combination with forward dynamics simulations, these nonlinear terms are counterproductive if simulated realistically, and require special treatment. Section 4.2.3 presents a new and simple technique to achieve a linearized behavior in operational space by dynamics-conditioning a virtual model.

4.2.1. Forward Dynamics Simulation

Forward dynamics simulation seeks to compute accelerations in the system's degrees of freedom in response to applied forces, and frequent synonyms are *direct dynamics* or simply *dynamics* [201]. In our scenario, we model robotic manipulators as systems of articulated, rigid bodies in revolute joint coordinates $\boldsymbol{q}$. The following set of ODEs describes the system's motion

$$\boldsymbol{\tau} + \boldsymbol{J}^T \boldsymbol{f}^n = \boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) \,, \tag{4.2}$$

in which $\boldsymbol{H}$ denotes the mechanism's positive definite joint space inertia matrix, $\boldsymbol{C}$ denotes Coriolis and centrifugal terms, and $\boldsymbol{G}$ holds the gravitational components. The dependency on $\boldsymbol{q}, \dot{\boldsymbol{q}}$ in the notation is omitted for brevity. We use Featherstone's notation [201] for the symbols but separate $\boldsymbol{G}$ and $\boldsymbol{f}^n$ from $\boldsymbol{C}$ for individual treatment in later sections. Torques in the robot's joints, if present, are summarized in the vector $\boldsymbol{\tau}$. Recall that the considered robots in this thesis do not offer a control interface on torque level (Section 2.1.1), and we set $\boldsymbol{\tau} \equiv \boldsymbol{0}$. What we do consider is the joint space effort $\boldsymbol{J}^T \boldsymbol{f}^n$, caused by the net force, encompassing all external loads at the end-effector.

For Inverse Kinematics, Section 2.1.4 highlighted the Jacobian transpose's inherent stability near singular configurations in comparison to the Jacobian Inverse. This advantage is naturally embedded in forward dynamics with Eq. (4.2).

Algorithms for Forward Dynamics

Algorithmic treatments of the equations of motion in computer programs go back to the mid-seventies [202], and consider the automatic derivation and integration of Eq. (4.2). Featherstone's more recent work [201] distinguishes two major categories in the field of forward dynamics algorithms: *propagation methods* and *inertia matrix methods*. The first category formulates joint accelerations and constraint forces locally, calculates their coefficients, and propagates them to neighboring bodies until local dynamics become solvable for each body. The dynamics are then likewise propagated to solve the complete tree [201]. One of the established algorithms to achieve this is the Articulated Body Algorithm (ABA) [203].

Inertia matrix methods as the second category take a three-step approach [201]:

(i) compute Coriolis terms $\boldsymbol{C}$ and gravity components $\boldsymbol{G}$ jointly as part of an inverse dynamics routine, such as the Recursive Newton-Euler Algorithm (RNEA) [204] by setting $\ddot{\boldsymbol{q}}, \boldsymbol{f}^n \equiv \boldsymbol{0}$
(ii) compute the joint space inertia matrix $\boldsymbol{H}$ e.g. with the Composite Rigid Body Algorithm (CRBA) [205], [206]
(iii) solve $\boldsymbol{H}\ddot{\boldsymbol{q}} = \boldsymbol{J}^T \boldsymbol{f}^n - \boldsymbol{C} - \boldsymbol{G}$ for the accelerations $\ddot{\boldsymbol{q}}$.

Methods from both categories are suitable in our concept and the choice depends primarily on practical considerations. Inertia matrix methods are faster for unbranched chains of up to eight revolute joints [201] and thus offer a slight performance benefit for our six-axis control case. After having separated the accelerations, the solution is obtained incrementally by numerical integration, which delivers $\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)$ for our motion-actuated robots.

4.2.2. Simplifications for Control

Closed-loop control requires computing the previous steps in real-time, and simplifications could enable faster feedback control cycles with better contact stability. To illustrate room for improvement, we first recapitulate the importance of dynamics for torque-actuated systems and then illustrate differences for motion-actuated systems: When the robot is at rest with $\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} = \boldsymbol{0}$, the influence of $\boldsymbol{H}$ and $\boldsymbol{C}$ vanishes and gravity alone accounts for the joint torques. During motion, the additional effect of inertia, Coriolis and Centrifugal terms gain importance and neglecting them results in high errors in the pre-computed torques $\boldsymbol{\tau}$ to drive the manipulator, which is difficult to compensate with feedback control [207], [208]. Ideal modeling of manipulator dynamics would lead to a response of a unit mass along each Cartesian degree of freedom [209]. Using inverse dynamics and good approximations of $\boldsymbol{H}, \boldsymbol{C}$ and $\boldsymbol{G}$ are thus crucial in achieving high tracking performance. Including the manipulator dynamics is common in two ways: In operational space [22], [210] and in joint space with usage of the RNEA for instance [211].

Motion-actuated robots, however, change this setting: Non-backdrivable transmissions in the joint servos realize gravity compensation at steady state, and high-gain feedback control compensates non-linear disturbances during fast motion independently for each joint. This changes the philosophy to using the dynamics of Eq. (4.2) and $\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{G}$ as parameters for motion generation. This has some resemblance to the DMPs from Section 2.2.6, in which a dynamical system is used for motion generation due to a non-linear disturbance. In this comparison, the net force $\boldsymbol{f}^n$ would take that role and drive the system, whose response is computed with numerical integration. The design of the dynamics parameters is thus arbitrary, and can actively be used to shape the desired behavioral characteristics. This supports our concept of using robot control to mimic the simulator's behavior we had during skill acquisition, and we propose two simplifications to aim for this goal:

(i) $\boldsymbol{G}(\boldsymbol{q}) \equiv \boldsymbol{0}$: The first simplification is to avoid gravitational drift in steady state. This assures that the virtual system does not sink to the ground and the net force has no need to constantly compensate this virtual load throughout varying joint configurations.

(ii) $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \equiv \boldsymbol{0}$: The second simplification is to discard velocity-related, non-linear influences on the robot's motion. The net force's effect shall be independent of the robot's joint velocity.

This reduces the complexity of forward dynamics to the computation and inversion of the joint space inertia matrix $\boldsymbol{H}$ and we have

$$\ddot{\boldsymbol{q}} = \boldsymbol{H}^{-1} \boldsymbol{J}^T \boldsymbol{f}^n \tag{4.3}$$

as an unbiased forward mapping of the net force to joint space accelerations. The next step is to decouple the six Cartesian control axes for the net force through conditioning $\boldsymbol{H}$ with mass and inertia distribution.

4.2.3. Manipulator Dynamics Decoupling

The mapping from task space to joint space with Eq. (4.3) makes use of the Jacobian transpose and thus inherits its beneficial stability. The Jacobian implicitly contains the robot's kinematics with the configuration of joint axes, their limits, and the lengths of the links in

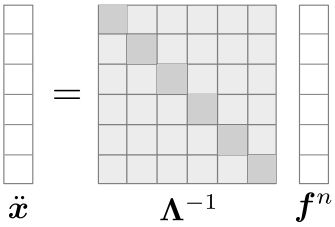


Figure 4.3.: Illustration of the ideal mapping matrix from wrench space to Cartesian acceleration. $\boldsymbol{\Lambda}$ is a 6×6 matrix for both redundant and non-redundant systems and should be diagonal for arbitrary joint configurations.

between. While the virtual system will reflect the real robot's kinematics, the joint space inertia matrix $\boldsymbol{H}$ is the primary interface for artificially shaping the dynamics of the virtual system. We use the CRBA algorithm [206] for its computation, which determines the elements H_{ij} based on composite inertias. Parameterizing its link masses and moments of inertia allows us to influence how the system ultimately accelerates in the direction pointed to by $\boldsymbol{f}^n$ after multiplication.

This response can be formulated in operational space, for which we follow Khatib's and Featherstone's derivation [212]: The time derivative of the end-effector's spatial velocity $\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}$ yields its acceleration with

$$\ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\ddot{\boldsymbol{q}}\,. \tag{4.4}$$

We focus on instantaneous accelerations and set velocity-dependent terms to zero. Substituting Eq. (4.3) leads to

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}\boldsymbol{H}^{-1}\boldsymbol{J}^T\boldsymbol{f}^n \tag{4.5}$$

for the Cartesian instantaneous acceleration of the virtual system due to the net force $\boldsymbol{f}^n$. In contrast to the manipulator Jacobian, the joint space inertia matrix is symmetric and positive definite, so that its inverse always exists. This leads to a stable mapping near singular configurations. Using Khatib's notation for this quantity we obtain

$$\ddot{\boldsymbol{x}} = \boldsymbol{\Lambda}^{-1}\boldsymbol{f}^n \tag{4.6}$$

for the mapping with $\boldsymbol{\Lambda} = \boldsymbol{J}^{-T}\boldsymbol{H}\boldsymbol{J}^{-1}$ being the joint space inertia matrix in operational space [22].

The goal of dynamics decoupling is to achieve a nearly constant, diagonal mapping with Eq. (4.6), independent of the robot's joint positions. Fig. 4.3 illustrates this desired behavior as matrix multiplication. Ideally, each component of $\boldsymbol{f}^n$ affects motion only in or along its respective axis. For realistic dynamics, this is however not the case. Since the robot is an articulated chain of joints and segments, a general motion in arbitrary Cartesian dimensions will affect several degrees of freedom. Consider the illustration from Fig. 4.4 (a). The net force causes linear and angular acceleration of the segments and in turn, perceives the system as composite rigid body inertia at the point of application. This overall quantity strongly depends on the joint configuration and introduces off-diagonal terms in $\boldsymbol{\Lambda}$ which distort the Cartesian response $\ddot{\boldsymbol{x}}$. This thesis contributes a simple design change to mitigate this effect.

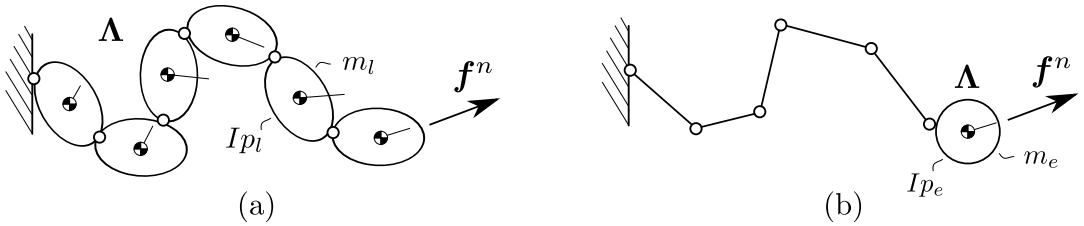


Figure 4.4.: (a) Virtual model for a 6-joint robot. The net force perceives the segments' moments of inertia as a configuration-dependent, composite rigid body inertia. (b) Making the end-effector's mass and inertia dominant shall linearize its dynamics in operational space.

Virtual End-Effector Dominance

The non-linearities are caused by the overall rigid body inertia of the segments' constellation, which changes dynamically during motion. The net force $\boldsymbol{f}^n$ itself will act on the virtual mechanism's end-effector as the point of application. This leads to a simple idea: To keep the composite inertia close to the point of attack for the net force, we can shape the virtual manipulator dynamics in such a way, that the end-effector's mass and inertia outweigh the other segments. If the virtual system's mass is concentrated in the point of application, the perceived composite inertia for the net force is mainly constant during motion and the manipulator would appear close to an idealized unit mass.

Fig. 4.4 (b) shows this concept. The center of mass stays with the end-effector and the operational space inertia $\boldsymbol{\Lambda}$ is constant across joint configurations. Other segments are reduced to kinematic links with insignificant own mass and inertia and thus have a vanishing influence on the overall dynamics.

We refer to this effect as *end-effector dominance* and propose

$$\gamma = \frac{m_e}{m_l} = \frac{Ip_e}{Ip_l}\,, \tag{4.7}$$

as a ratio for designing how much the end-effector's mass m_e and polar moment of inertia Ip_e outweigh the other links' dynamics m_l and Ip_l, respectively. Section 5.2.2 of the validation shows empirically that increasing γ leads to linearized, decoupled behavior for Cartesian closed-loop control.

4.3. Forward Dynamics Compliance Control

So far, the virtually conditioned model moves with simplified forward dynamics simulation according to an external net force. The next step is to design a closed-loop controller around this method that can regulate the setpoints of both the skill's predictions and the nominal assembly motion: Section 4.3.1 presents the control scheme in form of a generalized compliant controller. Section 4.3.2 details its algorithmic implementation and discusses two variants and their implications concerning time integration. Section 4.3.3 highlights two important specializations that cover frequent applications: The tracking of reference motion in free space, and pure force control, where forces and torques guide the robot without restoring forces. Finally, Section 4.3.4 summarizes how this type of controller is combined with the skill models from Chapter 3 to create human-inspired controllers for robotic assembly tasks.

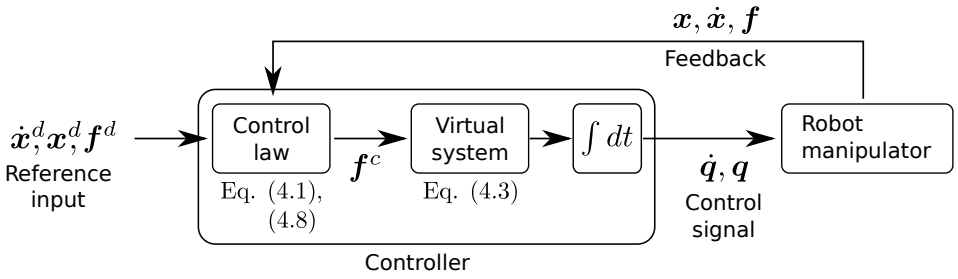


Figure 4.5.: Closed-loop compliant control with forward dynamics. Feedback is closed around a PD-controlled net force that enables motion and force setpoint tracking. A linearized, virtual system simulates reference joint motion that becomes the control signal for the real robot.

4.3.1. Control Scheme

The net force is the primary interface for control and the goal is to decrease the individual components' error, i.e. finding an equilibrium between the setpoints and current state. This means (i) decreasing the error between target pose and current end-effector pose, (ii) matching the target velocity and (iii) tracking the reference force profile when in contact.

Focussing motion-actuated systems with non-backdrivable inner position control loop poses the inherent challenge of contact stability. Adding additional control gains adds partial redundancy to the stiffness and damping parameters. This redundancy, however, allows to separate quasi-static stiffness from error rejection and introduces an additional parameter for reactiveness to mitigate contact stability issues.

Control Law

We use a PD regulator in the form of

$$\boldsymbol{f}^c = \boldsymbol{K}_P \boldsymbol{f}^n + \boldsymbol{K}_D \dot{\boldsymbol{f}}^n \tag{4.8}$$

with positive semi-definite gain matrices $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$. An explicit integral gain is not required to eliminate steady state errors. The virtual system has an integral part of itself during forward dynamics simulation from constant $\boldsymbol{f}^c$ to joint accelerations $\ddot{\boldsymbol{q}}$ and further to joint velocities $\dot{\boldsymbol{q}}$ and positions $\boldsymbol{q}$, respectively.

Adding gains to the overall net force $\boldsymbol{f}^n$ treats the individual errors the same. Like in Admittance Control and in contrast to Parallel Force/Position control, there is no prevalence of the force error over motion error. During control, the real dynamics of the manipulator, especially configuration-dependent structural stiffness of the inner joint position control loop, will have an influence on stability in contact. This structural stiffness will vary in different joint configurations and we can, therefore, not completely avoid $\boldsymbol{K}_P, \boldsymbol{K}_D$ being partly task and robot dependent. However, the controller contributes the linearization approach of the virtual system's dynamics from Section 4.2.3 and benefits constant control gains while the robot operates in different joint configurations.

Closed-Loop Control

Fig. 4.5 shows the control law embedded into a closed-loop control scheme. The reference input comprises $\boldsymbol{x}^d, \dot{\boldsymbol{x}}^d$ from the nominal trajectory, and $\boldsymbol{f}^d$ as the skill's predicted strategies. Feedback from the real manipulator closes the loop and feeds into the net force as the superposition of individual errors. A wrist force-torque sensor measures the contact wrench $\boldsymbol{f}$ and $\boldsymbol{x}, \dot{\boldsymbol{x}}$ are obtained via suitable Forward Kinematics (FK) routines. In each iteration, the virtual system accelerates in response to the controlled $\boldsymbol{f}^c$, and delivers, after time integration, the control signal for the real manipulator, which is the control plant of the scheme. The virtual model is considered part of the controller. Its role is that of a forward model, a proxy that computes reference motion, which becomes the control signal for the real plant. We repeat that this control concept does not require nor use the real robot's dynamics model. Instead, the virtual system is dynamically linearized with an unrealistic mass distribution to effect a linear acceleration response in operational space.

4.3.2. Implementation

Algorithm 1 shows the closed-loop control's steps. The pseudo-code should be implemented in a performant programming language. Depending on the manipulator used, the OEM driver for the inner joint control often runs between $100\,\mathrm{Hz}$ and $1\,\mathrm{KHz}$ and typically requires control signals $\boldsymbol{q}, \dot{\boldsymbol{q}}$ under real-time requirements.

Algorithm 1 Forward dynamics compliant control

1: **procedure** CONTROL LOOP($\boldsymbol{x}^d, \dot{\boldsymbol{x}}^d, \boldsymbol{f}^d, \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}$)
2: $\quad \boldsymbol{x} \leftarrow \mathrm{FK}(\boldsymbol{q})$
3: $\quad \dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}$
4: $\quad \Delta\boldsymbol{R} = \boldsymbol{R}^d\boldsymbol{R}^{-1}$
5: $\quad [r_x, r_y, r_z]^T \leftarrow \Delta\boldsymbol{R}$
6: $\quad \Delta\boldsymbol{x} = \boldsymbol{x}^d - \boldsymbol{x} = [\Delta x, \Delta y, \Delta z, r_x, r_y, r_z]^T$
7: $\quad \Delta\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}^d - \dot{\boldsymbol{x}} = [\Delta\dot{x}, \Delta\dot{y}, \Delta\dot{z}, \Delta\omega_x, \Delta\omega_y, \Delta\omega_z]^T$
8: $\quad \boldsymbol{f}^n = \boldsymbol{f}^d - \boldsymbol{f} + \boldsymbol{K}\Delta\boldsymbol{x} + \boldsymbol{D}\Delta\dot{\boldsymbol{x}}$
9: $\quad \boldsymbol{f}^c = \boldsymbol{K}_P\boldsymbol{f}^n + \boldsymbol{K}_D\dot{\boldsymbol{f}}^n$
10: $\quad \boldsymbol{H} \leftarrow$ CRBA[206] $\quad$ with $m_e, Ip_e = 1, \quad m_l, Ip_l = 1/\gamma$
11: $\quad \ddot{\boldsymbol{q}} = \boldsymbol{H}^{-1}\boldsymbol{J}^T\boldsymbol{f}^c$
12: $\quad \dot{\boldsymbol{q}} = \dot{\boldsymbol{q}} + \ddot{\boldsymbol{q}}\Delta t$
13: $\quad \boldsymbol{q} = \boldsymbol{q} + \dot{\boldsymbol{q}}\Delta t$
14: $\quad$ **return** $\boldsymbol{q}, \dot{\boldsymbol{q}}$
15: **end procedure**

The algorithm assumes $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ via measurements from the manipulator's joint sensors and $\boldsymbol{f}$ from a wrist-mounted force-torque sensor. It further assumes that a suitable routine is available for the Forward Kinematics (FK) and that the manipulator Jacobian $\boldsymbol{J}$ has been freshly computed in each loop.

The computation of the rotation difference is done in step 4, in which $\boldsymbol{R}^d, \boldsymbol{R} \in \mathbb{R}^3$ are the rotation matrices of the target pose and the current pose, respectively. We derive the Euler-Rodrigues vector components from that difference matrix in step 5. They represent the difference between target and actual state in axis-angle notation, where $\boldsymbol{r} = [r_x, r_y, r_z]^T$ sets the rotation axis and its norm $\|\boldsymbol{r}\|$ is the rotation angle. Since the

incrementally running control loop faces small relative rotation differences in each run, the known axis-angle ambiguities at $\pm\pi$ do not occur. The differences of translational and angular velocities from step 7 can be computed element-wise.

Note that the joint space inertia matrix $\boldsymbol{H}$ from step 10 requires setting the virtual link masses and inertias before its computation. After setting these parameters with a suitable end-effector dominance of e.g. $\gamma = 100$, $\boldsymbol{H}$ can be computed according to a fixed scheme [206]. This needs to be done in each loop due to its explicit dependency on the joint positions. Its inverse is guaranteed to exist for step 11 due to being positive-definite. The last steps 12-13 for time integration offer two alternatives which we briefly discuss in more detail.

Time Integration

One strength of the forward dynamics-based approach is that we obtain control signals on the acceleration level, i.e. the virtual system responds with joint accelerations. Depending on the joint control interface of the real robot, we have single-time integration or double integration, respectively. Since integrators are low-pass filters, they lead to smooth, delay-free control signals. This contrasts with controllers that generate less smooth control signals and may require additional filtering.

Note that Δt can be arbitrarily chosen. It represents a virtual simulation time and is partially redundant with the control gains. But in comparison to the proportional gains, its influence is quadratic on the delta of joint positions $\Delta \boldsymbol{q}$ between the loops. It is, therefore, simpler to set it to a fixed value and not expose it as a control parameter.

Considering position-controlled robot interfaces, we have two options for the time integration of these signals: (i) accumulating velocity and (ii) starting with instantaneous accelerations from zero in each loop. While the first approach leads to a discretized, dynamical system that is always in motion, the second one can be understood as a gradient-based solver with a numerical step width. Both have advantages and disadvantages and are more or less suitable for different use cases. Table 4.2 and Table 4.3 show the different features.

Table 4.2.: Instantaneous acceleration during time integration

scheme: $\dot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}\Delta t, \quad \dot{\boldsymbol{q}}_{\text{last}} \equiv \boldsymbol{0}$
+ No explicit damping required for free space motion
+ No velocity build-up and no overshooting pose targets
− Proximity-to-goal dependent solver convergence
− No crossing of singularities, because inertia is not conserved
− Noisy joint velocity control signals

4.3.3. Control Applications

Forward Dynamics Compliance Control (FDCC) offers a generalized scheme for the inclusion of both reference trajectories and reference force-torque profiles. The net force and its modular superposition of components offer application-dependent specializations. We highlight two simplifications that cover frequent use cases.

Table 4.3.: Accumulating velocity during time integration

scheme: $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\text{last}} + \ddot{\boldsymbol{q}}\Delta t$
+ Double integration leads to smoother signals
+ Smooth joint velocity control signals
+ Inertia can overcome singularities (elbow flip)
− The problem of overshooting pose targets
− Redundant manipulators need additional nullspace damping

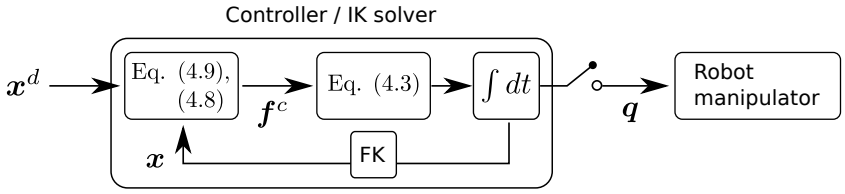


Figure 4.6.: Control scheme for the special case of sampled motion tracking.

Sampled Motion Tracking

Consider motion in free space, for which no interaction with the environment is anticipated. These trajectories might move the parts into place before the actual assembly. High autonomy in this setting demands flexibility of the executions, e.g. by incorporating dynamic localization of workpieces. Being reactive in operational space requires the controller to follow ad-hoc targets $\boldsymbol{x}^d(t)$ and the reference velocity $\dot{\boldsymbol{x}}^d(t)$ is of secondary interest. FDCC covers this by simplifying the net force to

$$\boldsymbol{f}^n = \boldsymbol{K}(\boldsymbol{x}^d - \boldsymbol{x}) \,. \tag{4.9}$$

Adding an internal feedback loop for the virtual system allows to transition from a conventional controller to an exact IK solver. Fig. 4.6 shows the according scheme. Robot control signals are passed to the robot open-loop at the desired rate, as indicated by the switch.

A requirement on joint control level often demands that the drivers receive continuous control signals. If the targets are sparsely sampled, the controller must adequately interpolate between them. This is where FDCC benefits from its linearized behavior in operational space that supports goal-directed motion with an arbitrary granularity of interpolation: The control gains together with the number of internal loops as additional parameters can be tweaked to adjust behavior between a smoothing controller towards an exact Inverse Kinematics solver. Section 5.2.7 evaluates this feature.

For underactuated systems, the principle of forward dynamics-based simulation implements a best-effort strategy: Instead of failing with impossible-to-reach targets, the controller gets as close as possible by simulating a pulling wrench until mechanical limits are reached. This is beneficial for underactuated systems because unfeasible targets are less intuitive and often arise during a teaching in different morphology such as teleoperation.

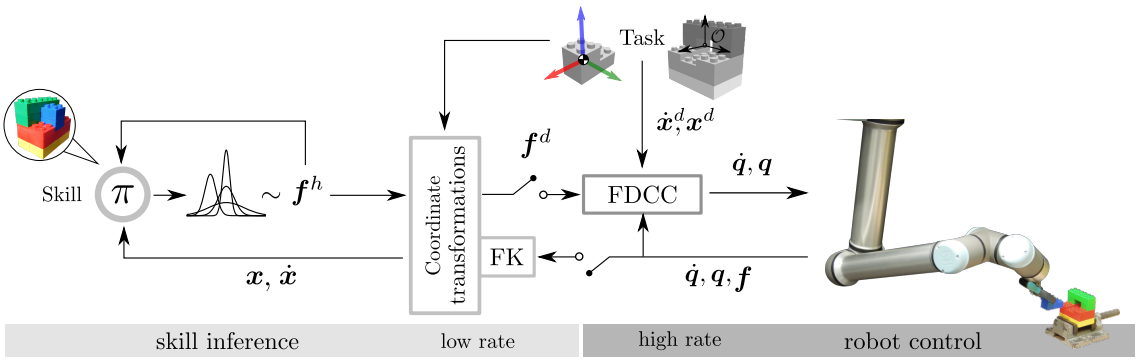


Figure 4.7.: Skill controllers for robotic assembly tasks.

Free Force Control

As an alternative to motion control, forces and torques can be used to steer the robot's end-effector. Contacts with the environment are anticipated in this case. For highly constrained tasks, a reference motion in open-loop playback can be counterproductive, such that restoring forces through virtual springs are dropped for this scenario. For assembly, the skill's strategies can freely guide the manipulator within search-like patterns. The according net force simplifies to

$$\boldsymbol{f}^n = \boldsymbol{f}^d - \boldsymbol{f} \tag{4.10}$$

with a behavior similar to that of the simulator for recording human behavior. Forces and torques steer the robot's end-effector and the control gains determine the responsiveness of the motion. The absence of a damping term requires to use the instantaneous time integration method, which in combination with the responsiveness mimics a velocity-proportional damping.

4.3.4. Skill Control

Fig. 4.7 shows the combination of skill models and FDCC into skill controllers. The overall control scheme has two cycles: Skill inference on the left and robot control on the right. The left cycle runs at comparatively low rates, at which the skill model predicts a mixture of distributions to sample from for control. This model is task-specific and is valid within the region of trained relative poses of the assembly parts. After coordinate transformations as described in Section 3.4.1, this prediction is passed to FDCC as one of several possible inputs. The nominal trajectory during assembly is optional. Whether following a target with a virtual stiffness is suitable depends on the use case at hand. FDCC from algorithm 1 computes suitable control signals for the robot's actuators. It abstracts the real manipulator and mimics the behavior of the simulator in moving the robot in response to the force-torque setpoints. This second cycle runs at a comparatively high rate, which is necessary to maintain the contact stability of the robot and its environment. Feedback of measured joint state signals and from the wrist-mounted force-torque sensor close this control loop. Forward kinematics and coordinate transformations feed that information back into the skill model for the next strategy prediction.

4.4. Summary and Conclusion

Skill controllers fulfill the transfer of assembly skills to different systems. Our concept continued the idea of forward dynamics simulation from the skill learning environment in form of a new mapping approach to admittance control. This mapping is particularly suitable for motion-actuated systems because it enables controllers to link end-effector wrenches naturally to joint accelerations with the benefit of smooth control signals after time integration. Another major benefit is the inherent stability near singularities that contrasts with approaches based on the Jacobian inverse. The designed control law builds upon integrating individual error components into a common net force, which drives a virtual model of the manipulator to generate reference motion for the real setup. Further simplifications to the dynamics of the virtual system and the contribution of end-effector mass dominance achieve a linearization of the control response in operational space. The result is a general forward dynamics compliance controller (FDCC) for the different assembly phases: Motion without contact, contact transition, and contact-dominated assembly with human-inspired skills as an overlay. The following chapter provides an experimental evaluation of the different key concepts and propositions.

5. Evaluation

Previous chapters described the theory from demonstrating assembly in simulation, learning human behavior into probabilistic forward models, and providing a way of executing these skills in form of force-torque strategies through robot control. This chapter now provides empirical evaluation for important aspects of each contribution: Section 5.1 evaluates the skill models and their architecture and how respective hyperparameters influence learning performance. These experiments are conducted in the assembly simulator. Section 5.2 then focuses on the new robot controller and investigates the characteristics of its mapping in comparison to suitable baselines. Some of these aspects can be evaluated empirically in simulation. Others, such as the different control interfaces for practical applications, are evaluated on real robotic manipulators. Section 5.3 finally evaluates the skill transfer to real hardware through the combination of both for two assembly tasks, providing a proof-of-concept for the methods proposed to program human-inspired skill controllers for robotic assembly.

5.1. Skill Learning Performance

Section 3 designed skills as neural network models that predict force-torque strategies for robot control during assembly operations. Input to these models were data streams of previous commands and object-relative motion, with the two primary components of sequential memory and ad-hoc modeling of Gaussian distributions. All aspects motivate further experimental investigations that are covered in the following sections:

- The dataset, used for evaluation. Section 5.1.1
- Tests and simulation setup. Section 5.1.2
- How the number of Gaussian kernels affect performance. Section 5.1.3
- The influence of sequential memory length. Section 5.1.4
- An ablative study for different input feature combinations. Section 5.1.5
- The influence of learning data quality on performance. Section 5.1.6

We use two measures for learning performance: The batch-wise loss function as described in Section 3.3.4 over the training iterations, and testing the models' assembly performance in simulation. The first shall illustrate the different models' ability to approximate the dataset as a whole and the second shall evaluate if the studied effect is also visible during regression. Using simulation in this regard allows a bigger number of executions and adds more statistical significance. Section 5.3 shows assembly skill transition to real hardware.

5.1.1. Dataset and Training

The assembly skills need data to learn from. Since this thesis introduced several new concepts for model architecture, handling of input features, and conventions about coordinate systems, there are no common datasets available. We continue the example from

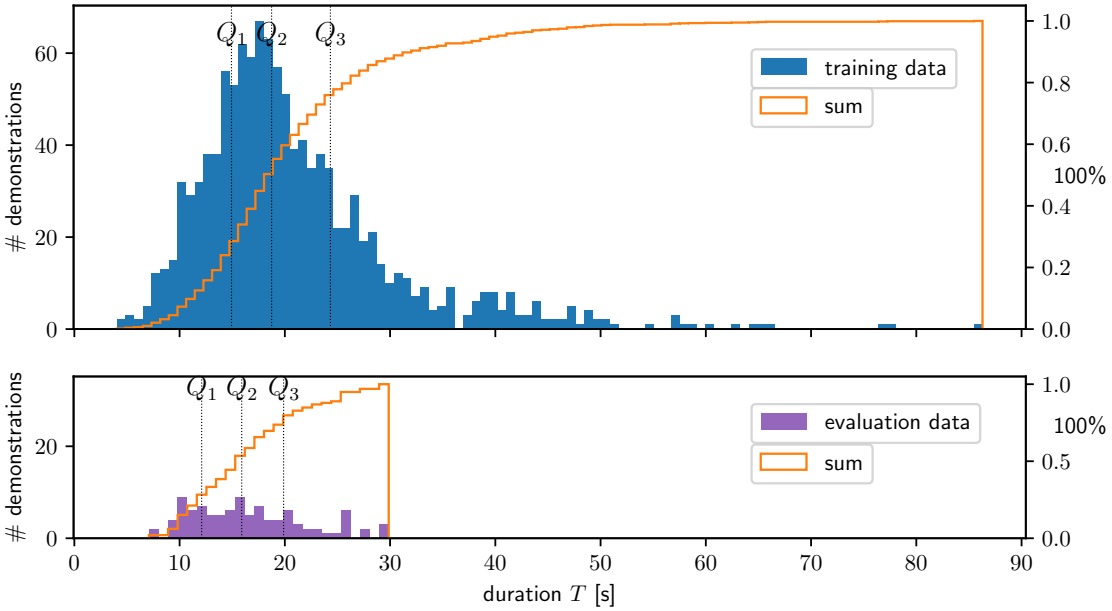


Figure 5.1.: Dataset for the toy assembly. The sets comprise 1155 and 101 demonstrations for training and evaluation, respectively. This is equivalent to approximately 7 hours and 10 minutes of performed assembly in the simulator, with the median at approximately 18 s. The input feature streams were recorded at 100 Hz, yielding an overall data size of 960 MB. The dashed, vertical lines with Q_1 to Q_3 indicate the quartiles that separate the datasets into equally populated quarters.

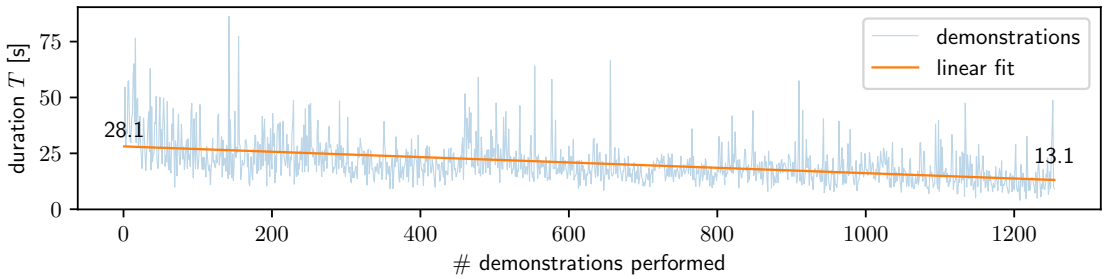


Figure 5.2.: Recorded demonstrations, ordered after the time of creation. Growing experience using the simulator decreases the average duration of assembly.

previous chapters and use a specifically created dataset for the toy assembly. Fig. 5.1 shows the composition into two datasets: The training data are used to update the learnable parameters of the skill models in providing inputs and labels for the computation of the error gradients. The evaluation data are used for the batch loss of the learning curves. These data are previously unknown to the models during learning and are excluded from the learnable parameter update. Note that the dataset is intentionally big to enable an in-depth investigation of the concepts and that a smaller set suffices for practical applications. The time for training the models was classically between $10\,\mathrm{min}$ to $1\,\mathrm{hour}$ on a modern CPU, mostly depending on the number k of Gaussian kernels and the length N of sequence memory. We did not consider speed improvements with highly parallelized implementations on GPUs, which can become promising for production. With few exceptions, trained models had 15.000 ± 1 % parameters in the experiments with a size of $150\,\mathrm{kB}$.

As described in Section 3.2.3, our concept includes a simple joystick-like 3D controller as teach device. Although being mostly intuitive to use, there is also a learning effect on the side of the teacher as shown in Fig. 5.2. Varying start positions of the assembly parts add noise, but the tendency shows that experience decreases the average duration of assembly. This underlines the method's applicability for industrial programmers and engineers used to such devices.

5.1.2. Tests in Simulation

Simulation played an essential role in acquiring assembly skills from human behavior. We now re-use the simulator and let the skill models repetitively solve the assembly task. Their force-torque predictions steer the active assembly object in the same way as was done when demonstrating with the teach device. Note that we artificially increased friction and stiction between the parts to make the simulated task presumably harder than the real system. A good performance in simulation is, therefore, an early proof-of-concept for a later transition to real robotic systems. We use the Euclidean distance to the goal pose $\|\boldsymbol{x}\|$ as a measure of progress, which vanishes for successful assemblies. Fig. 5.3 shows one such curve for successful skill execution. Two plateaus indicate challenging spots for the skills: The first is slightly over $10\,\mathrm{mm}$ and the second is around $5\,\mathrm{mm}$ to the goal, respectively. Part jamming occurs everywhere along the path and needs to be taken care of.

Testing the assembly skills in simulation is bounded by CPU performance. The models

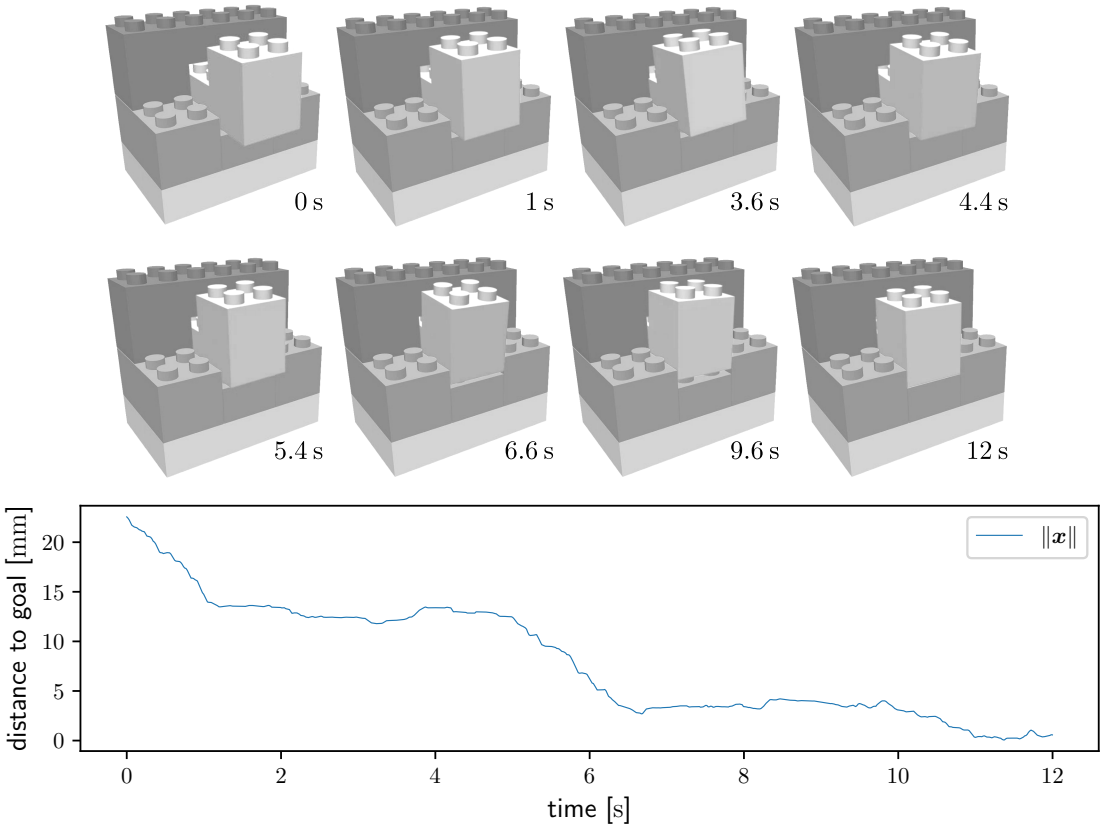


Figure 5.3.: A successful skill execution in simulation. The Euclidean distance to the goal shows assembly progress and serves as a criterion for completeness. Plateaus indicate part jamming that the skill needs to solve.

currently predict at maximum rates between $12\,\mathrm{Hz}$ to $20\,\mathrm{Hz}$, depending on model complexity. We slowed the simulator down to 20 % real-time to replicate similar conditions as during recording human behavior at $100\,\mathrm{Hz}$. The following experiments use varying starting poses in the middle of the toy assembly with a focus on intermediate jamming.

5.1.3. Mixture Densities

The data analysis from Section 3.2.4 showed that the skills need to learn distributions instead of the Mean Squared Error for capturing the dataset's characteristics, and Section 3.3.2 developed a probabilistic output with MDNs for this purpose. This experiment investigates the number of Gaussian kernels as hyperparameter and their influence on learning performance. We adjusted the number of nodes in the LSTM component for each model to obtain an equal amount of trainable parameters despite different numbers of Gaussian kernels in the output layer and to make the models comparable. We applied dropout [213] to stabilize training and to prevent the models from overfitting. Two exceptions were necessary: The models with $k = 1$ and $k = 2$ required a higher amount of trainable parameters to at least partially learn the dataset and finally converge. All models were trained with a sequence memory of $N = 25$ steps. Fig. 5.4 (*Top*) shows the results. Each iteration of the learning curves is one weight update on the complete mini-batch. The curves show that more Gaussian kernels improve the models' ability to learn the dataset and succeed in resolving possible ambiguity among the labels. A saturation is visible after $k = 8$, after which learning does not significantly improve.

Fig. 5.4 (*Bottom*) shows the results of assembly tests in the simulation. We use the boxplot statistic to allow a quick overview of the distribution: They group the results into quarters, with the mean separating the upper and lower 50 %, and the whiskers representing upper and lower 25 %, respectively. Outliers are marked with the $+$ symbol. The tests underline that a model with only one Gaussian kernel is not able to predict meaningful force-torque strategies. Only models with more Gaussian kernels succeed in the assembly task in simulation. In contrast to what the learning curves indicate, $k = 4$ performs best on this test. Note, that the given time to solve the task was restricted to $12\,\mathrm{s}$ simulated time. The last region for jamming the assembly is around $5\,\mathrm{mm}$ to the goal. Skills that surpass this distance are likely to finish successfully if given more time.

In conclusion, skill models require a mixture of distributions to learn datasets of complex manipulation and to predict successful force-torque strategies in simulation. An optimum of Gaussian kernels will partially depend on the complexity of each dataset.

5.1.4. Sequence Memory

Memory allows the skill models to learn sequential patterns in data and Section 3.3.1 developed a sequential model around the LSTM. This experiment investigates the number of unfolding steps N in BPTT during training as a hyperparameter and its effect on learning performance. In contrast to the number of Gaussian kernels in the MDN component, sequence length in the LSTM does not change the trainable parameter count of the neural network. We chose $k = 8$ Gaussian kernels for all models. In combination with the recorded dataset at $100\,\mathrm{Hz}$, the considered sequence lengths span from $10\,\mathrm{ms}$ ($N = 1$) to $5\,\mathrm{s}$ ($N = 500$), which is the time window the skill models can oversee for making predictions during training. Note that all models had an infinite time horizon during testing in

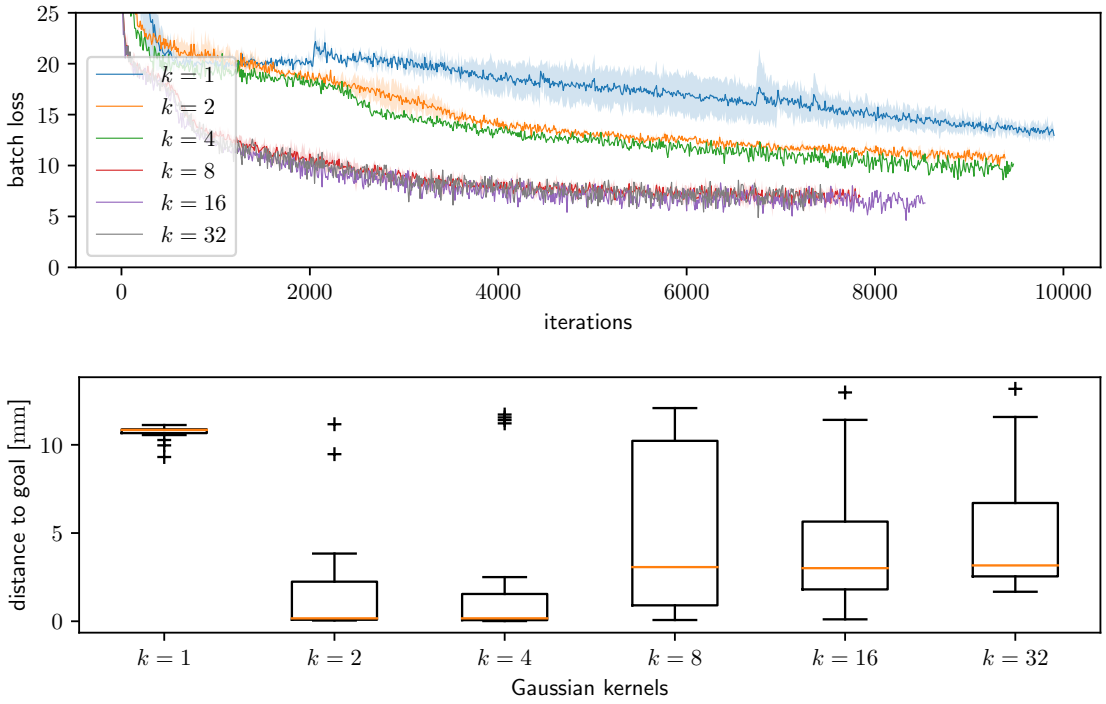


Figure 5.4.: Effect of Gaussian kernels on skill learning performance. *Top*: Learning curves for the different models with $50 \cdot 10^3$ ($k = 1$), $30 \cdot 10^3$ ($k = 2$) and $15 \cdot 10^3$ ($k = 4 \ldots 32$) learnable parameters. All models were trained with a minibatch size of 128 and a learning rate of $5 \cdot 10^{-4}$. *Bottom*: Results of assembly tests in simulation. The boxplots show the distance to the goal after 12 s simulation time. Each model was tested 20 times with the start configuration from Fig. 5.3.

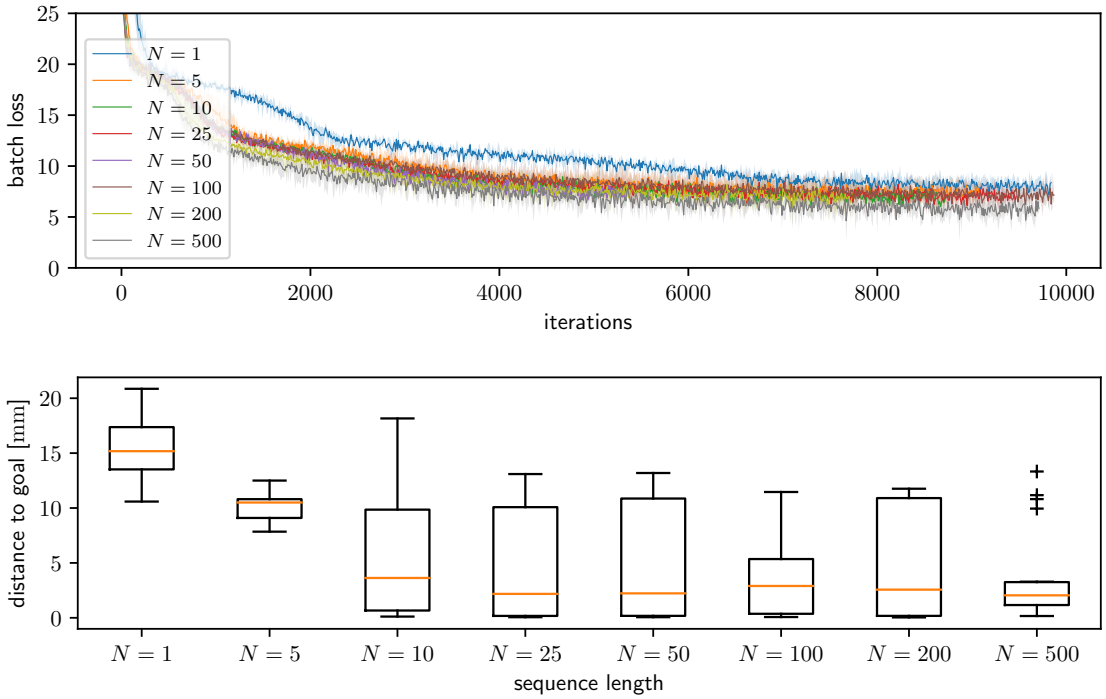


Figure 5.5.: Effect of sequence memory on skill learning performance. *Top*: Learning curves for the different models, each with $15 \cdot 10^3$ parameters. N describes the number of unfolding steps in BPTT. All models were trained with a minibatch size of 128 and a learning rate of $5 \cdot 10^{-4}$. *Bottom*: Performance in simulation. Each model was tested 20 times with the assembly start configuration from Fig. 5.3.

simulation, i.e. all models continuously updated their cell and hidden state throughout the complete assembly.

Fig. 5.5 (*Top*) shows the respective learning curves. A higher sequence length improves the skill models' ability to approximate the dataset. This effect, however, is less pronounced after a minimal sequence length is surpassed, and the middle field indicates a similar performance. An exception occurs with $N = 500$, whose model provides the lowest evaluation loss. Fig. 5.5 (*Bottom*) shows the assembly tests in simulation. The results confirm that a minimal sequential memory during training is required for the skills to be partially successful. In accordance with the learning curves, $N = 500$ performs best on this test.

In conclusion, skill models must possess the ability to learn from sequential data. Being less obvious in learning curves alone, a minimum length is required to predict coherent and successful force-torque strategies during inference. Depending on the dataset at hand, longer sequences can further improve performance.

5.1.5. Input Feature Ablation Study

Basing our concept for assembly skills on mental forward models, Section 3.1 proposed extracting and learning strategies from the observed data stream $\boldsymbol{f}^h, \boldsymbol{x}, \dot{\boldsymbol{x}}$. The following experiment evaluates this proposition and compares the learning performance to that of

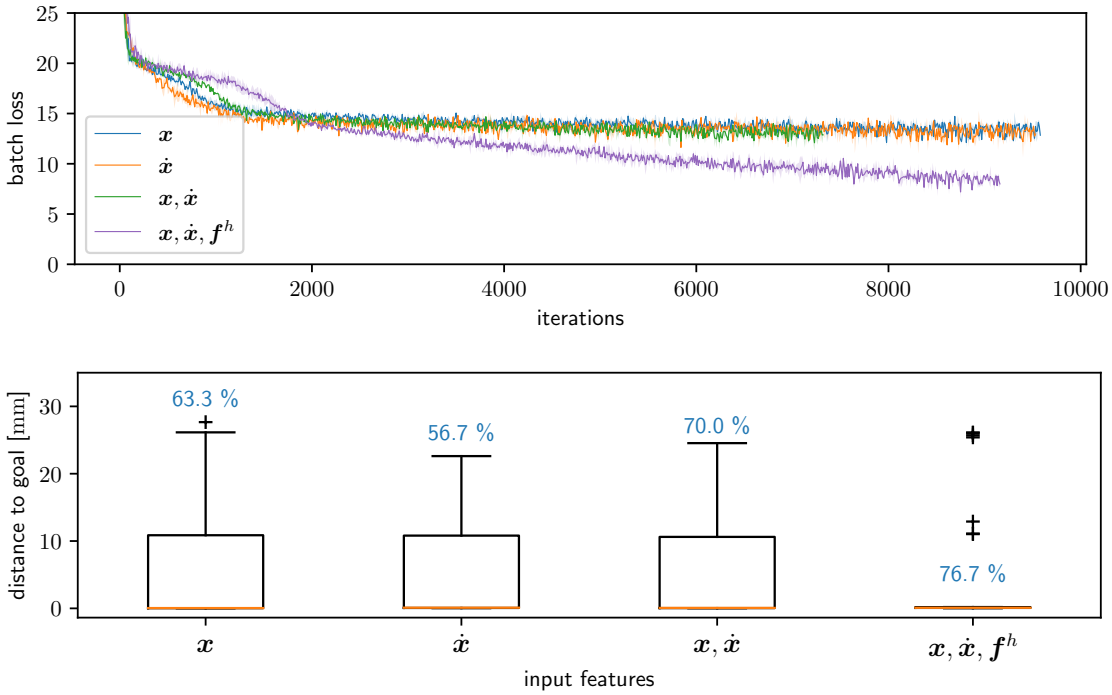


Figure 5.6.: Input features and skill learning performance. *Top*: Learning curves of the different models, all trained with $k = 4$, $N = 25$, learning rate $5 \cdot 10^{-4}$ and minibatch size 128. *Bottom*: Assembly performance in simulation. 30 starting poses of mixed difficulty were evaluated alike for each model. Successful executions reached the goal within $0.5\,\mathrm{mm}$ in $20\,\mathrm{s}$. The overall success rate is depicted on top of each boxplot and includes outliers.

models with fewer input features during training and inference. The rest of their neural network architecture with sequential memory and Gaussian mixtures stays unchanged, and they continue to predict force-torque strategies for control like in previous experiments.

We consider the following input feature combinations and test their ability to ground assembly skills:

- $\boldsymbol{x}$: Relative pose to test if quasi-static changes between parts are sufficient
- $\dot{\boldsymbol{x}}$: Relative motion to investigate if models can learn without an absolute reference
- $\boldsymbol{x}, \dot{\boldsymbol{x}}$: The combination of both to test for synergies
- $\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{f}^h$: The full human behavior to test the effect of including past actions

Fig. 5.6 (*Top*) shows the learning curves. There is no obvious difference between pose, velocity, and their combination on their ability to approximate the dataset. In contrast, having the stream of human behavior as input lets the model predict force-torque outputs that agree much better with the dataset, as the lower evaluation loss shows. However, tests in simulation are required for this case, because this effect could be due to the model learning to just repeat the last seen action. Fig. 5.6 (*Bottom*) shows the results of 30 assemblies each. The relative pose is already a good input feature for grounding assembly skills. Interestingly, the model trained on relative velocity $\dot{\boldsymbol{x}}$ alone also produces good results. 56 % of the executions are successful. The combination of the two performs

better, as expected, but all models got stuck towards the goal. The model trained on all three input feature vectors proofs to be more robust against jamming, such that failures are considered as outliers by the boxplot algorithm.

In conclusion, the skill architecture with sequential memory is capable of learning strategies also on reduced input features. The best performance is achieved when grounding skills in the interplay between sensed state $\boldsymbol{x}, \dot{\boldsymbol{x}}$, and motor command $\boldsymbol{f}^h$.

5.1.6. Demonstration Quality

Data-driven approaches classically depend on the quality of their training data. This experiment investigates if and how the quality of demonstrations influences skill learning performance. All demonstrations for the datasets were recorded until finishing the assembly in simulation and considered successful examples. We use their duration as a measure of quality. This implies that the shorter an assembly took during the demonstration in the simulator, the better is the assumed quality of the recorded labels for strategies. On average this shall provide a sufficient concise measure despite varying start positions. We used the quartiles Q_1 - Q_3 from Fig. 5.1 to separate the training set into quarters and obtain five datasets to train models for investigation:

- $0 \rightarrow Q_1$: Demonstrations that took up to $14.93\,\mathrm{s}$
- $Q_1 \rightarrow Q_2$: Demonstrations that took between $14.93\,\mathrm{s}$ and $18.74\,\mathrm{s}$
- $Q_2 \rightarrow Q_3$: Demonstrations that took between $18.74\,\mathrm{s}$ and $24.32\,\mathrm{s}$
- $Q_3 \rightarrow \infty$: Demonstrations that took longer than $24.32\,\mathrm{s}$
- $0 \rightarrow \infty$: All demonstrations

We further used the first quartile on the evaluation set for computing the batch loss during training. This includes only demonstrations up to $12.09\,\mathrm{s}$ and shall evaluate if the models predict high-quality labels during training iterations. Fig. 5.7 *Top*) shows the learning curves. Most models perform similarly in predicting the high-quality labels of the evaluation set. The model trained on the last quarter is a little off at the end of the given training iterations. The results of the simulation test from Fig. 5.7 *Bottom*) shows this degraded performance. Note that all tests were cut off at $12\,\mathrm{s}$, marking the upper bound of the time to completion and equaling failure. The model trained on the complete dataset has the best performance. Interestingly, the model trained on the second quarter of the dataset has a similar performance with only 25 % training data. An explanation for this effect could be that this excerpt from the complete dataset provides a good trade-off between data quality and the amount of recorded time for training.

In conclusion, the quality of demonstration influences the skills' ability to learn successful force-torque strategies. Shorter and thus more goal-directed demonstrations can partially replace bigger datasets.

5.1.7. Conclusion

Deep neural network models are usually difficult to evaluate analytically. The previous sections, therefore, took an experimental approach to evaluate the individual components of our assembly skills. We chose the toy assembly as a challenging task, for which we created a specific dataset that was used to investigate learning curves and assembly tests in simulation. As a result, assembly skill models require both components of our

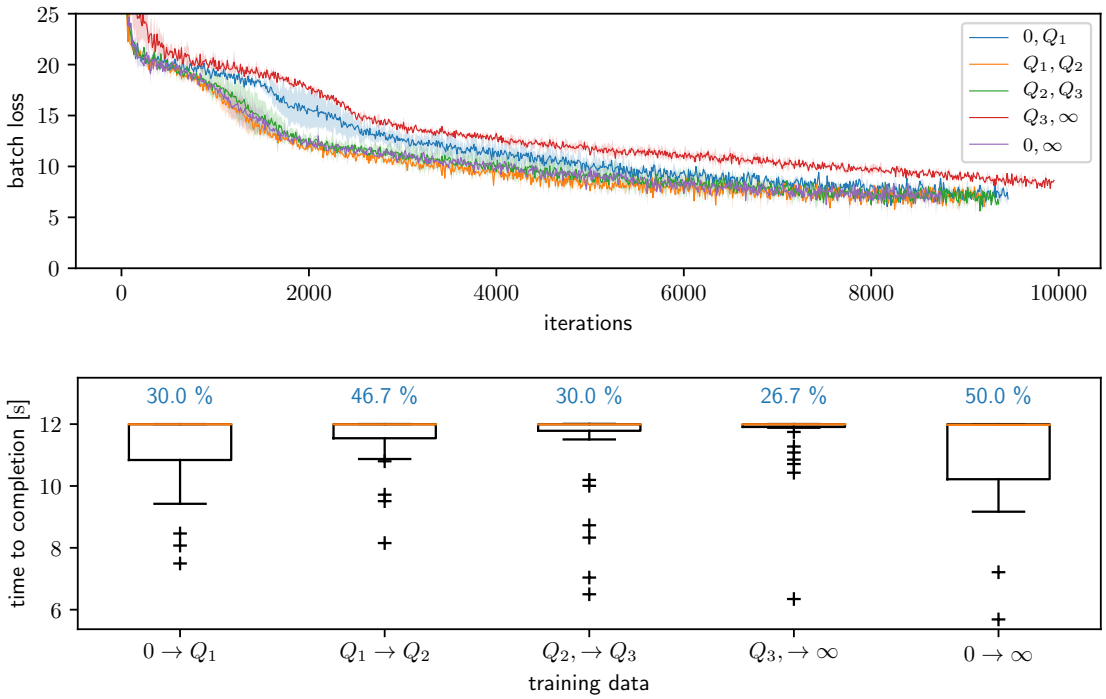


Figure 5.7.: Influence of data quality on skill learning performance. *Top*: Learning curves of the different models, all trained with $k = 4$, $N = 25$, learning rate $5 \cdot 10^{-4}$, and a minibatch size of 128. *Bottom*: Assembly performance in simulation. 30 trials with the start pose from Fig. 5.3 were tested for each model. Successful executions reached the goal of $0.5\,\mathrm{mm}$ in 12s. The overall success rate is depicted on top of each boxplot and includes outliers.

concept for predicting successful force-torque strategies against part jamming and tilting: An internal short-term memory for processing and reacting to sequential inputs and a mixture of probability densities to sample from for control. An ablative study further confirmed that assembly skills are best grounded in the interplay of motor commands and motion feedback. Finally, skill learning improves with the quality of demonstration data, which shows the method's suitability for programmers used to the teach device.

5.2. Robot Controller Performance

The new control concept of Forward Dynamics Compliance Control (FDCC) touched on several aspects that motivate an in-depth investigation. It builds around virtual forward models of the robots with a mapping from task-wrench space to joint-acceleration space as a crucial component. Characteristics of this mapping are investigated with a focus on linearized behavior in operational space and robustness in singular joint configurations. Both shall compare the overall performance of the controller's IK approach to suitable baselines. These experiments are performed in simulation. The last two experiments then focus on control performance on real robotic platforms.

Several experiments and results have been published in prior own work [214], [215], [216]. The following sections reiterate these result and embed them along with additional discussions in the bigger context of this thesis:

- Experimental setup and baselines. Section 5.2.1
- Effectiveness of dynamics linearization in operational space. Section 5.2.2
- Robustness in singular joint configurations. Section 5.2.3
- Comparison in evading ill-conditioning. Section 5.2.4
- Performance trade-off between manipulability and stability. Section 5.2.5
- Computational efficiency in comparison. Section 5.2.6
- Sampled motion tracking with a focus on sparse targets. Section 5.2.7
- Free force control with a focus on singular joint configurations. Section 5.2.8
- Compliance control with force tracking during motion in contact. Section 5.2.9

5.2.1. Experimental Setup

We use three robots during the evaluation in the following sections:

- Universal Robots UR10[1]
- Schunk Powerball LWA4P[2]
- KUKA KR16[3]

The footnotes indicate drivers and kinematics specifications provided by the ROS framework that we used. All robots are six-axis industrial robots, the UR10 and LWA4P belong to the lightweight category. The first experiments investigate various characteristics for the mapping matrix of the controller in theory, and can thus be carried out in simulation. This allows for rigorous testing of a big number of joint configurations and a better statistical significance. We chose the UR10 for these experiments due to its widespread usage

[1] https://github.com/ros-industrial/universal_robot
[2] https://github.com/ipa320/schunk_modular_robotics
[3] https://github.com/ros-industrial/kuka_experimental

Abbr.	$\ddot{\boldsymbol{q}} = \square\, \boldsymbol{f}^c$	$\ddot{\boldsymbol{x}} = \square\, \boldsymbol{f}^c$
JI	$\boldsymbol{J}^{-1}$	$\boldsymbol{J}\boldsymbol{J}^{-1}$
JT	$\boldsymbol{J}^T$	$\boldsymbol{J}\boldsymbol{J}^T$
DLS	$(\boldsymbol{J}^T\boldsymbol{J} + \alpha^2\boldsymbol{I})^{-1}\boldsymbol{J}^T$	$\boldsymbol{J}(\boldsymbol{J}^T\boldsymbol{J} + \alpha^2\boldsymbol{I})^{-1}\boldsymbol{J}^T$
FD	$\boldsymbol{H}^{-1}\boldsymbol{J}^T$	$\boldsymbol{J}\boldsymbol{H}^{-1}\boldsymbol{J}^T$
	(a)	(b)

Figure 5.8.: Mapping matrices for the experiments. The abbreviations stand for Jacobian inverse (JI), Jacobian transpose (JT), Damped Least Squares (DLS), and Forward Dynamics (FD). Two types are investigated: (a) Mappings from Cartesian space to joint space and (b) mappings from Cartesian space to Cartesian space. Caption and image from [214].

both in industry and academia. The last two experiments then use the real platforms - the LWA4P for free force control, and the UR10 and KR16 for compliance control. Table 5.1 shows the robots' specifications.

Table 5.1.: Robotic manipulators

Robot	Max. payload [kg]	Weight [kg]	Reach [mm]	Repeatability [mm]	Control frequency [Hz]
UR10	10	17	1300	0.1	125
LWA4P	6	15	730	0.15	100
KR16	16	235	1611	0.05	83.3

Mapping Matrices and Baselines

All cartesian controllers must ultimately generate control signals for the robots' joints. On the way to achieving this they solve IK, for which Section 2.1.4 reviewed current methods. The goal is to determine how our new approach compares to these baselines. We thus evaluate our forward dynamics-based approach (FD) against the Damped Least Squares (DLS) and against the two edge cases Jacobian inverse (JI) and plain Jacobian transpose (JT). The last one is of particular interest because our new approach only differs by the pre-multiplication of the inverse of the conditioned joint space inertia matrix $\boldsymbol{H}^{-1}$.

Fig. 5.8 lists the different mapping matrices. Depending on the experiment, we are either interested in the joint motion or the operational space motion they generate. They are implemented as displayed in C++ without further performance enhancements. The algorithms for computing $\boldsymbol{J}$ and $\boldsymbol{H}$ are supported by the Kinematics and Dynamics Library (KDL)[4].

The following values are chosen for the FD mapping:

$$m_e = 1\,\mathrm{kg}, \quad m_l = \frac{m_e}{\gamma}, \quad ip_e = 1\,\mathrm{kg/m^2}, \quad ip_l = \frac{ip_e}{\gamma} \tag{5.1}$$

[4]https://github.com/orocos/orocos_kinematics_dynamics

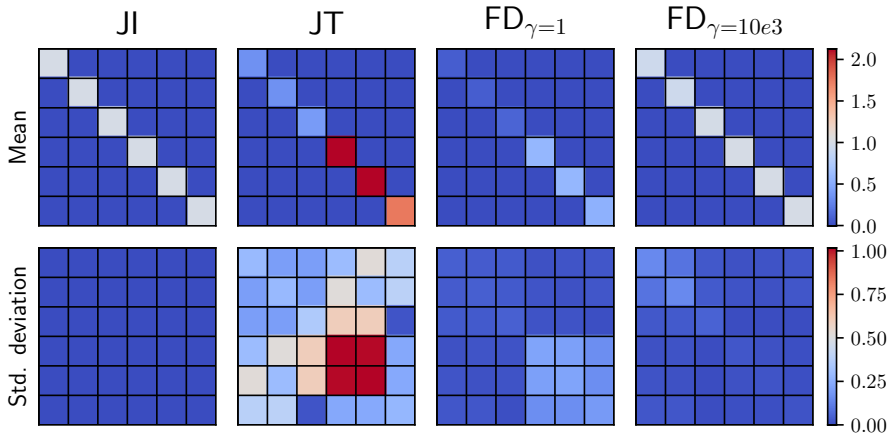


Figure 5.9.: Analysis of the 6×6 mapping matrices from Fig. 5.8 (b) for random joint configurations. The figures show the mean and standard deviation for the mappings in form of 2d heat plots, where the Jacobian inverse (JI) represents ideal behavior. The individual squares show the multiplicative gains from control wrenches to operational space acceleration for each of the six Cartesian dimensions. 100.000 joint configurations were sampled uniformly with $q_i \in [-\pi, \pi]$ for the statistics. Image from [214].

The ratio of virtual end-effector dominance γ is varied according to the investigation of each experiment.

5.2.2. Dynamics Linearization

This experiment shall evaluate the effectiveness of linearizing the virtual model dynamics in operational space. The mapping matrices are those of type (b) from Fig. 5.8, i.e. they compute how the end-effector accelerates with a wrench from the control law. The Jacobian inverse serves as a reference for ideal behavior, it is with $\boldsymbol{J}\boldsymbol{J}^{-1}$ equal to the identity matrix and can assure an ideal decoupling of non-linear, joint configuration-dependent terms everywhere in the robot's workspace. At this point, there are no suitable assumptions yet about the workspace of the robot, nor about possible or likely postures the robot might operate in. We thus test the effect of linearization for the controller on average with a big amount of random joint configurations and standard deviation as a statistic.

Fig. 5.9 shows the results of the analysis. The Jacobian inverse is ideal as expected: There are no off-diagonal gains that influence the individual Cartesian dimensions for the mapping and the vanishing standard deviation proofs its significance for the entirety of sampled joint configurations. Perfect diagonality means for the controller in this case that each dimension of a control wrench with this mapping only generates an acceleration along or about this axis. The results show that all mean matrices are diagonal. This underlines that there are no dominant cross-correlations on average between individual Cartesian dimensions for this robot's kinematics. However, the standard deviations show that configuration dependency does occur, especially for the Jacobian transpose. The red 2×2 square of its plot indicates a strong interdependency for the Cartesian angular x, y-axes. This mapping is thus suboptimal for the controller. In comparison, the forward dynamics-based approach achieves better results, even with a virtual end-effector mass

dominance of $\gamma = 1$. This shows that the multiplication of $\boldsymbol{H}^{-1}$ in the mapping already has a beneficial effect on decoupling the Cartesian axes for control. The results further show that with a significant end-effector mass dominance of $\gamma = 10e^3$, the forward dynamics mapping converges to the behavior of the ideal Jacobian inverse, which proofs the intended effect of linearization.

5.2.3. Singularity Robustness

Robustness to singular joint configurations makes controllers more generally applicable. Although regions with singular configurations can be pre-computed for industrial settings, singularities may arise ad-hoc during motion that is not planned, e.g. during teleoperation. The goal of this section is a qualitative investigation of how the forward dynamics-based mapping behaves near singularities.

Section 2.1.4 described Singular Value Decomposition (SVD) as a suitable method to investigate the manipulator Jacobian under rank deficiency. In our case, we apply this method to the mapping matrices from Fig. 5.8 (a), which we factorize according to

$$\text{mapping} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \quad \text{with } \Sigma_{ii} = \sigma_i. \tag{5.2}$$

The individual $\sigma_i \geq 0$ of the diagonal matrix $\boldsymbol{\Sigma}$ are the singular values. In contrast to the orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}^T$, which constitute rotations, they determine the scale of the mapping. For our experiments, σ_{min} and σ_{max} are of particular interest. They show the maximal and minimal scale for this specific joint configuration and illustrate how the mapping scales the Cartesian control signal to joint space. A vanishing σ_{min} means that control about some Cartesian axis gets lost during the mapping. Conversely, an exploding σ_{max} means that some Cartesian axis will cause extreme joint motion.

In this experiment, we drive the robot virtually from one arbitrary singular configuration into another with linear joint space interpolation. The resolution is 1000 steps for the complete path. In each step, we compute the respective mapping matrix and its singular values. Fig. 5.10 shows the curves of σ_{min} and σ_{max} between vertical, dashed lines.

$\boldsymbol{J}^{-1}$ marks the upper line in the top plot. Its σ_{min} is zero only at exactly singularities, meaning that no Cartesian dimension gets lost in the mapping. But its σ_{max} strongly increases towards both ends of the path, making controllers with this mapping unstable with unbounded outputs. The Jacobian transpose in contrast delivers stable, bounded outputs, but with a strongly degraded σ_{min} towards the singular configurations. The curves of the forward dynamics-based mappings show how σ_{max} stays in stable ranges, while at the same time approaching the ideal $\boldsymbol{J}^{-1}$. This effect is more pronounced with increasing virtual end-effector dominance γ and shows that a suitable trade-off is possible between the two corner cases.

5.2.4. Ill-Conditioned Configurations

Featherstone's investigation of the joint space inertia matrix showed ill-conditioning to be an inevitable phenomenon of mechanisms in general [217]. It manifests itself with high sensitivity to changes in inputs and makes it difficult to simulate or control the mechanisms [217]. He used

$$\kappa = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \tag{5.3}$$

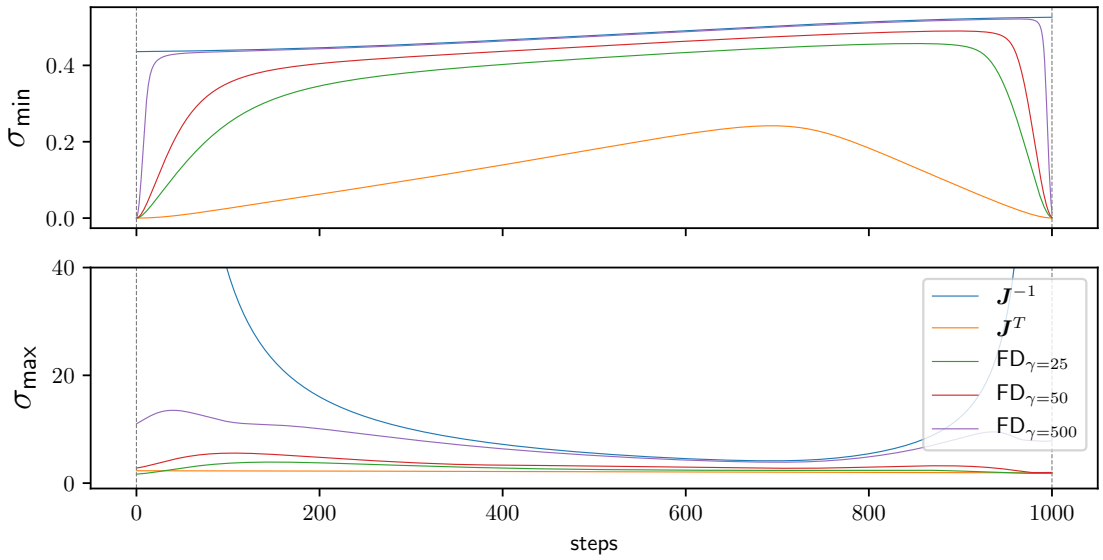


Figure 5.10.: Qualitative stability analysis near singular configurations. To obtain the curves, we interpolated linearly in joint space between two exemplary singular configurations and computed the mapping matrix according to Fig. 5.8 (a) for each step. The extremal singular values $\sigma_{\text{min}}, \sigma_{\text{max}}$ illustrate how forward dynamics achieves a trade-off between two corner cases through varying end-effector mass dominance γ.

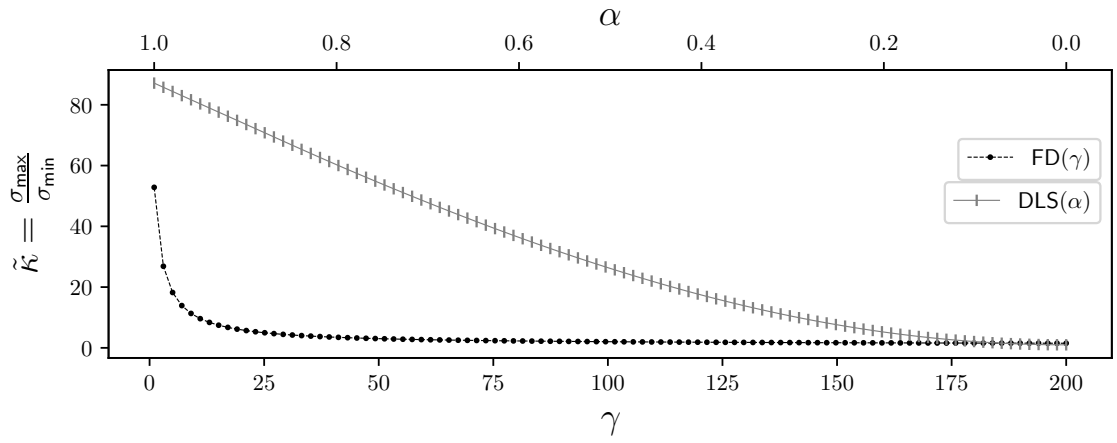


Figure 5.11.: Effect of ill-conditioning for the DLS and FD method under varying parameters. Each data point evaluates 1000 randomly sampled joint configurations. Image from [214].

as a measure and showed that high numbers drastically degrade performance. The goal of this experiment is to investigate the FD mapping under this criterion. We choose the DLS method in comparison due to being an established alternative to the plain Jacobian inverse in admittance controllers. Both methods change their behavior with γ and α, respectively, and allow for a parameter study.

We proceeded as follows: First, we computed a set of 1000 random joint positions by sampling uniformly from $[-\pi, \pi]$ for each joint. We then varied γ and α in discrete steps in their own parameter spaces and computed $\sigma_{\text{min}}, \sigma_{\text{max}}$ with SVD for their mapping matrices from Fig. 5.8(b) for the complete set of samples. This set also included near singular configurations by chance. Since numbers for ill-conditioning become huge near singularities due to $\sigma_{\text{min}} \to 0$, we used quartiles on the obtained κ and excluded these cases as outliers. Each point in the curves from Fig. 5.11 thus represents the median $\tilde{\kappa}$ as a measure of ill-conditioning.

The result shows that FD has a fast convergence in its own parameter space. Controllers with this mapping can effectively evade ill-conditioning already with approximately $\gamma = 75$. The DLS method converges slower and a suitable controller must operate with relatively low damping to reach these low values of ill-conditioning.

5.2.5. Stability and Manipulability

This experiment further investigates FD's performance concerning stability and manipulability and provides an empirical analysis of the workspace in comparison to the DLS method. Both stability and manipulability are concurrent features for controllers. As discussed in Fig. 5.10 from Section 5.2.3, the Jacobian Inverse marks the ideal corner case for manipulability, keeping a non-zero σ_{min} everywhere with a sharp drop to zero at exactly hitting singular configurations. In contrast, the Jacobian transpose noticeably decreases already in the neighborhood of singular configurations. Its σ_{max} curve, however, stays bounded and represents our corner case for ideal stability.

Similar to Fig. 5.11, the goal is a parameter analysis of DLS' damping term α and FD's end-effector dominance γ, but now focusing near-singular as the most challenging configurations. Additionally, we define two criteria,

$$\text{relative manipulability} := \frac{\sigma_{\text{min}}(\dots)}{\sigma_{\text{min}}(\boldsymbol{J}^{-1})}\,, \tag{5.4}$$

which measures how a mapping matrix maintains its manipulability in comparison to the ideal behavior of the Jacobian inverse, and

$$\text{relative instability} := \frac{\sigma_{\text{max}}(\dots)}{\sigma_{\text{max}}(\boldsymbol{J}^{T})}\,, \tag{5.5}$$

which compares the instability of the mapping to that of the Jacobian transpose.

To find a big amount of suitable configurations, we first sampled random joint positions as seeds and then used Particle Swarm Optimization (PSO) [218], [219] as a heuristic search method to converge to singular, and thus configurations of vanishing manipulability. PSO iteratively samples a given function and derives directions for local and global search. Using Yoshikawa's manipulability measure $|\det(\boldsymbol{J})|$ for non-redundant mechanisms [40] was faster to compute than searching for vanishing σ_{min} with SVD.

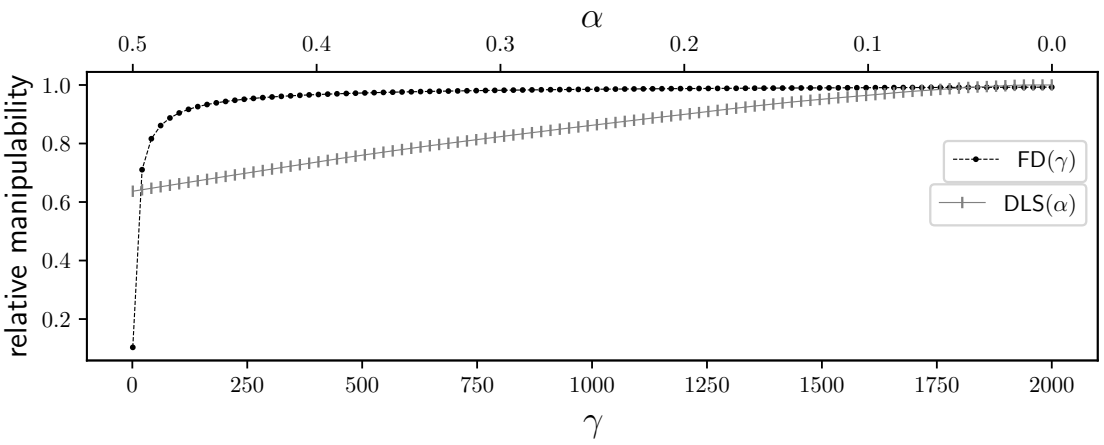


Figure 5.12.: Relative manipulability for the DLS and FD method in comparison to $\boldsymbol{J}^{-1}$ as an ideal reference. Each data point evaluates 1000 near-singular joint configurations. The FD method can be tuned to behave like the Jacobian inverse with increasing end-effector dominance γ. Image from [214].

For creating the diagrams, we discretized α and γ and evaluated the average of Eq. (5.4) and Eq. (5.5) for the mapping matrices from Fig. 5.8(a) on 1000 previously computed, near-singular configurations.

Fig. 5.12 shows the results for relative manipulability. Both DLS and FD reach the manipulability of the Jacobian inverse. This means that both mappings can be tuned to beware controllers near singular configurations from losing manipulability, i.e. loss of control for at least one Cartesian dimension. Note that this is the expected behavior of the DLS method which defaults to the Jacobian inverse for $\alpha = 0$ as a corner case. Interestingly, FD shows a strong gain in manipulability already with slightly increasing the end-effector dominance γ.

Fig. 5.13 shows the results for relative instability and the fundamental difference between FD and DLS. The latter maintains better relative stability throughout most of the observed parameter space. However, towards matching the Jacobian inverse for high manipulability, instability explodes as expected and approaches infinity. FD in contrast stays bounded and can operate with high manipulability near singular configurations, making it a safe alternative to DLS.

5.2.6. Computational Efficiency

The analysis so far showed good performance of the FD mapping regarding linearity of dynamics, stability, manipulability, and the phenomenon of ill-conditioning. Practical implementations into control schemes need fast execution times of this mapping, which poses an additional requirement for performance. This experiment shall compare the FD mapping to alternatives from Fig. 5.8(a). Note that the plain Jacobian transpose is an unusual candidate for admittance control. It is nevertheless useful in highlighting the cost of the additional multiplication with $\boldsymbol{H}^{-1}$ of the FD method. We also included Selectively Damped Least Squares (SDLS) [44] as an established method for comparison. The method increases usability over DLS by replacing the parameter α with automatic, σ_{ii}-specific damping. This algorithm works component-wise and has no direct formulation

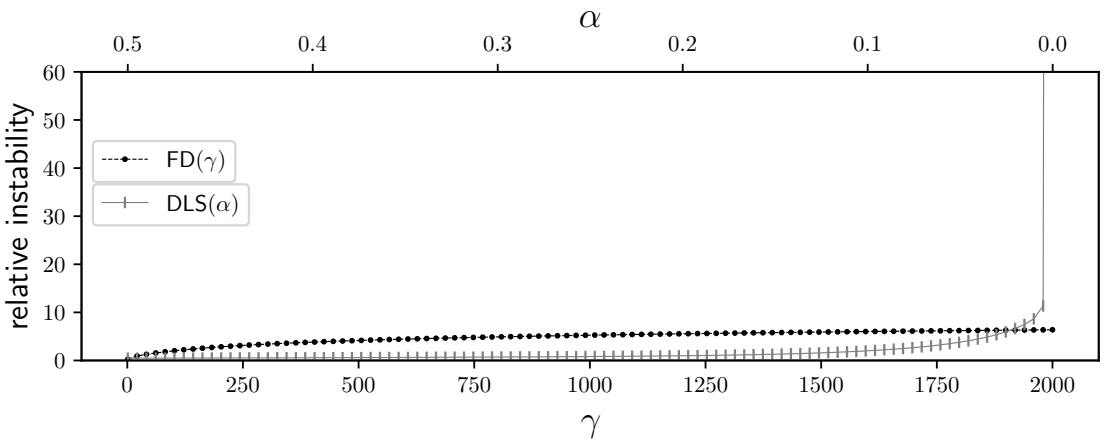


Figure 5.13.: Relative instability for the DLS and FD method in comparison to $\boldsymbol{J}^T$. Each data point evaluates 1000 near-singular joint configurations. The FD method stays bounded when operating at high manipulability. Image from [214].

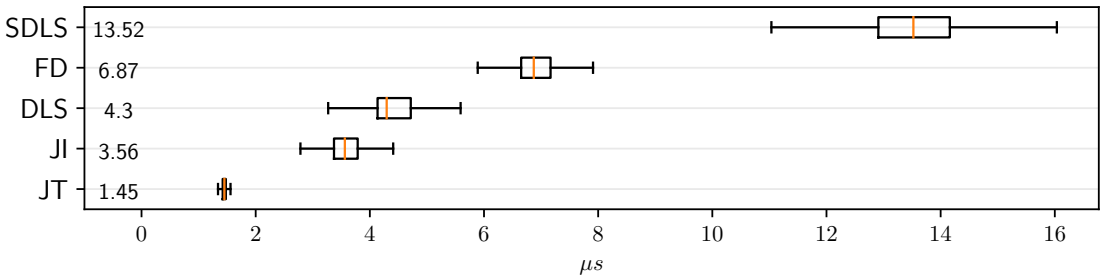


Figure 5.14.: Execution times of different mapping approaches on an Intel® Core™ i7-4900MQ with non-real-time OS. Reproduced from [214].

as a mapping matrix. But it likewise allows measuring the cost of computing a vector of joint commands from a Cartesian input.

To estimate the cost of each method, we computed the vector of instantaneous joint accelerations $\ddot{\boldsymbol{q}}$ with a fictitious, constant input $\boldsymbol{f}^c = \mathbf{1}$. The start state $\boldsymbol{q}$ of the robot's joints was randomly sampled but kept identical during one test of each method. Fig. 5.14 shows the performance in execution time for 10^5 tests. Vertical lines indicate the median of the boxplots, respectively. Fluctuation in execution times is due to the non-real-time operating system used, the algorithms themselves are deterministic.

The results show that the FD method around virtual forward dynamics simulations is still computationally tractable: The method ranks in the middle of common approaches. The execution time of only a few μs underlines its suitability for real-time closed-loop control.

5.2.7. Motion Tracking

A common task for robot control is to track nominal trajectories before establishing contact with the environment. We consider a special form of task space motion in the fol-

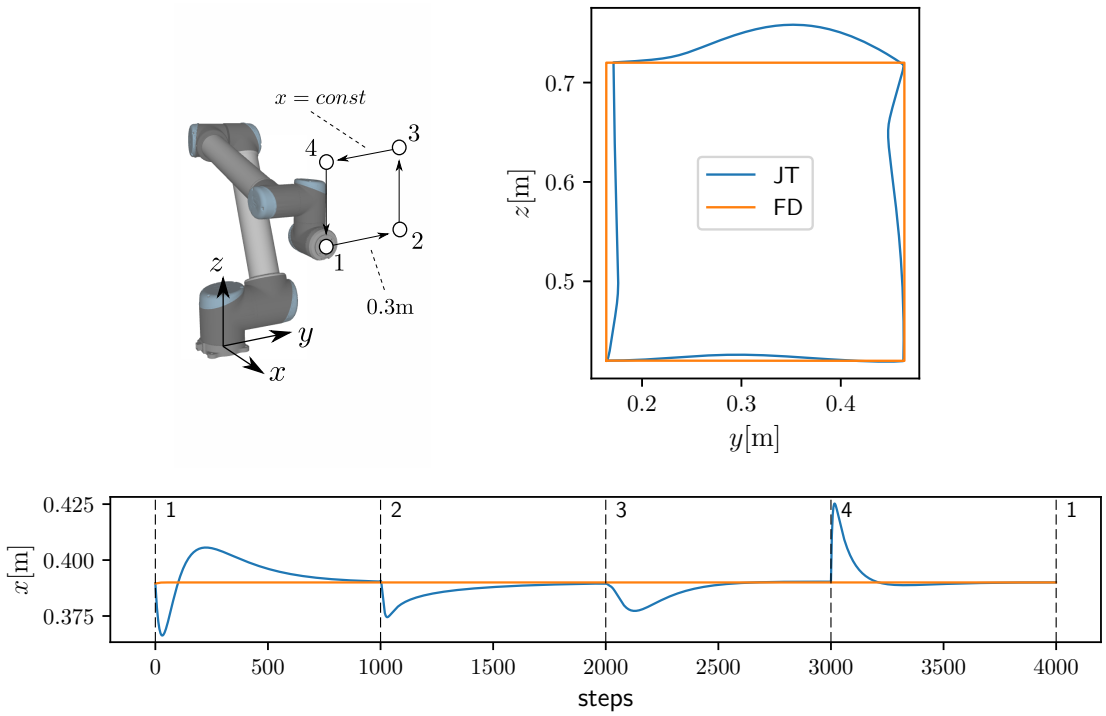


Figure 5.15.: Analysis of path accuracy during interpolation to the corners of a square. Both controllers used $\Delta t = 0.1\,\mathrm{s}$ as the virtual time constant and 50 solver iterations for each step. The traveled end-effector positions are computed with the robot's forward kinematics. Reproduced and adapted from [215].

lowing two experiments: Ad-hoc sampled trajectories that are described with discrete target poses at specific time stamps. Computing IK for these targets and passing the joint configuration directly to the actuators provokes infeasible jumps. Instead, the controllers need to interpolate to provide smooth control signals for the inner joint position control loops of the robot.

Sparse Waypoints

This experiment evaluates FDCC's special case of pure motion control from Fig. 4.6 and its ability to provide goal-directed interpolation. Both the FD and the JT mapping are tested in this controller scheme for comparison.

The experiment consists of interpolating to the four corners of a square in front of the robot and plotting the traveled end-effector positions via forward kinematics for inspection. Fig. 5.15 shows the starting configuration and the results. We computed 1000 intermediate steps towards each next counter-clockwise corner, taking $\boldsymbol{x}^d$ as a fixed target. The JT method leads to distorted paths in Cartesian space and indicates that this controller requires densely sampled trajectories to not branch out during interpolation. In contrast, the controller with the FD method shows goal-directed interpolation, which proofs the effectiveness of the task space linearization of Section 4.2.3 for control applications.

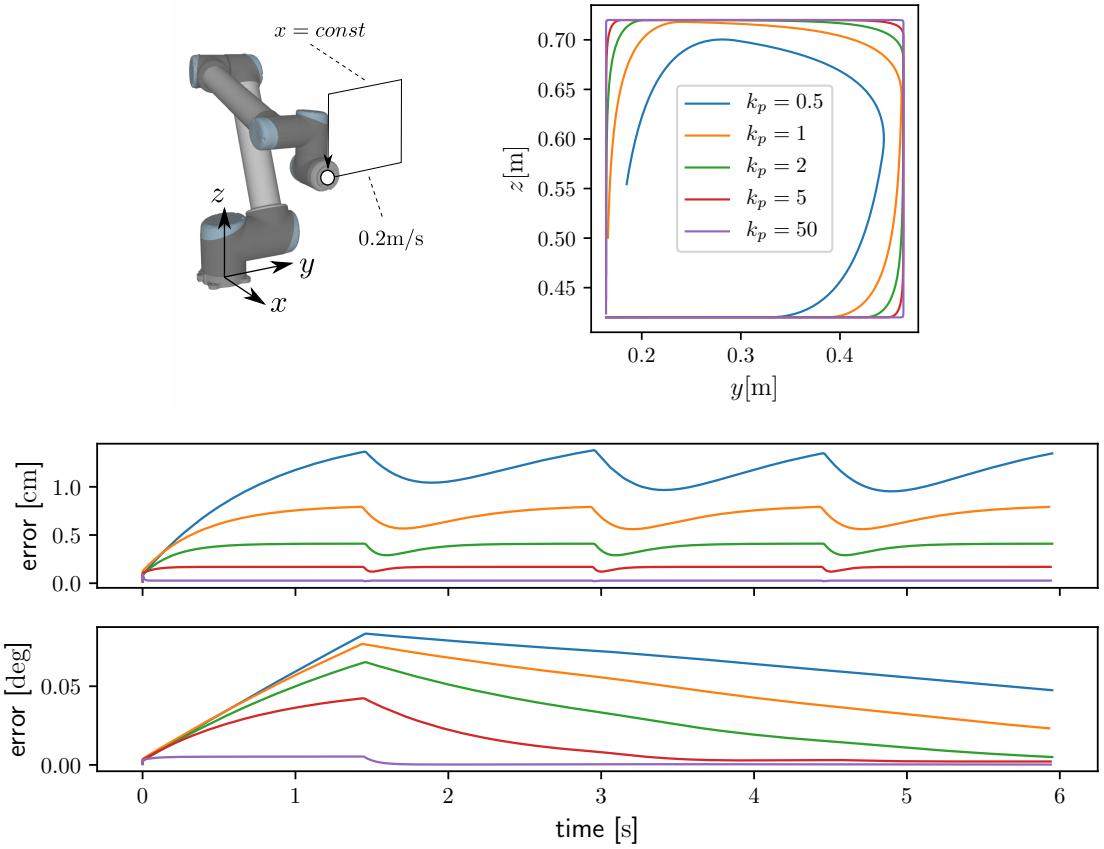


Figure 5.16.: Analysis of tracking performance for a square-moving target of $0.2\,\mathrm{m/s}$. The family of curves is computed by varying k_p in the controller's gain matrix $\boldsymbol{K}_p = k_p \cdot \mathrm{diag}([1, 1, 1,\ 0.1, 0.1, 0.1])$. We chose $\Delta t = 0.1\,\mathrm{s}$ as the virtual time constant for integration and 10 internal solver iterations. *Top*: The end effector's traveled position in the task space. *Bottom*: The respective translational and rotational errors to the moving target. Reproduced and adapted from [215].

Exactness vs Smoothness

The next experiment shows how the controller allows to continuously fade to an IK solver for exact target tracking. We create a reference motion that follows the square from the previous experiment corner-by-corner at a constant speed of $0.2\,\mathrm{m/s}$. During the execution, we sample the moving target with a frequency of $100\,\mathrm{Hz}$.

Fig. 5.16 shows that the controller can transition to an exact IK solver for higher controller gains. Applications may exploit this feature to find good trade-offs between noise filtering and target following.

5.2.8. Force Control

Assembly skills predict force-torque strategies that guide the robot's end-effector during assembly. This experiment evaluates the free force control component of FDCC with

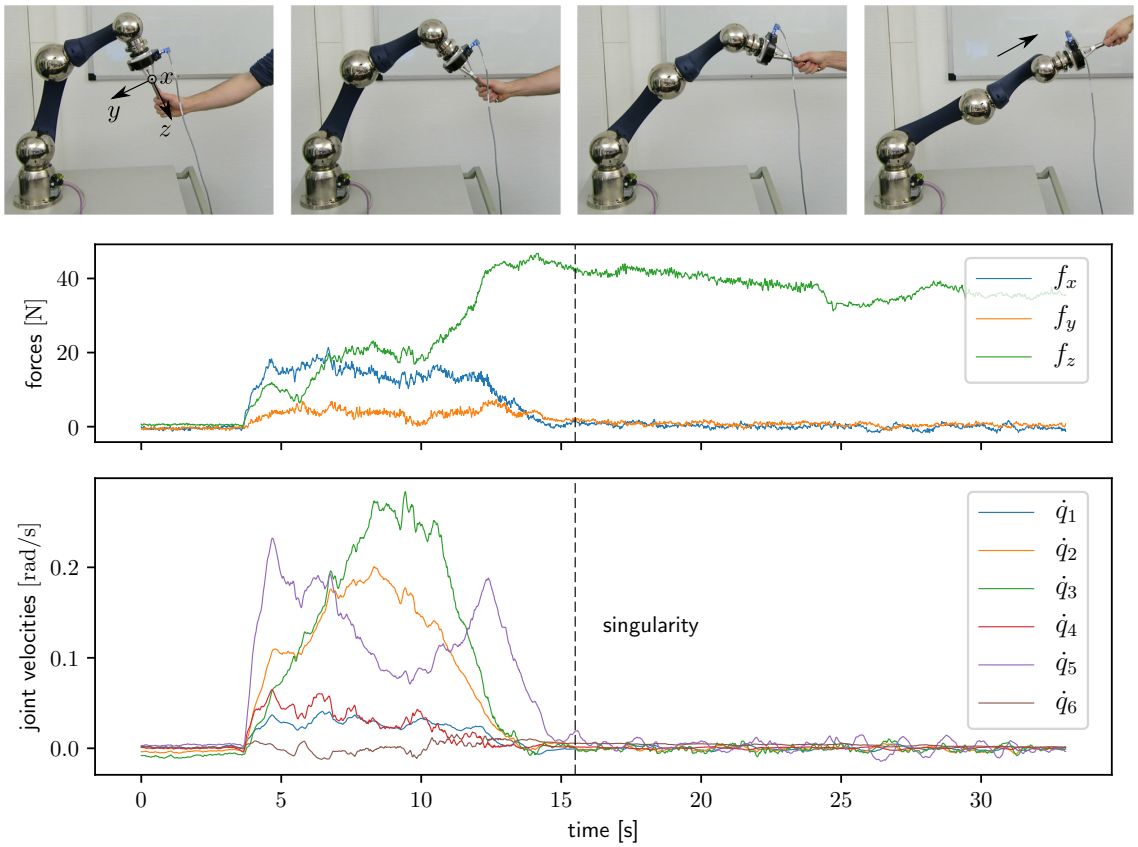


Figure 5.17.: Free force control in singularity. *Top*: The Schunk LWA4P manipulator with a wrist-mounted force-torque sensor that measures externally applied forces. A handle is attached to guide the manipulator. The robot is pulled into a singular joint configuration at reaching its workspace limits. *Bottom*: Applied end-effector forces and computed control signals to the LWA4P's actuators. The forward dynamics-based controller maintains the robot stable in singular configurations. Reproduced and adapted from [216].

a focus on behavior in singular joint configurations. The setup consists of the Schunk LWA4P robot with a wrist-mounted force-torque sensor and an attached handle. Stiffness is set to zero and the robot can be guided freely in task space. Fig. 5.17 (*Top*) shows how the robot is pulled to its maximum reach. This test comprises both the singularity of the workspace's limits and the singularity through the alignment of several joints. Fig. 5.17 (*Bottom*) shows the applied end-effector forces and the according joint command signals: The manipulator behaves stable in the singularity at reaching its workspace limits, which is indicated on the time axis with a vertical, dashed line. After this point, the commanded velocities are still smooth and bounded, despite external forces continue to act on the end-effector. This experiment used the accumulating time integration method from Table 4.3 which allows using momentum to flip the robot's elbow. The alternative instantaneous method can be used if this option is not desired in an application.

In conclusion, force control proofs to be stable in singular configurations and thus best supports assembly skills in guiding the robot.

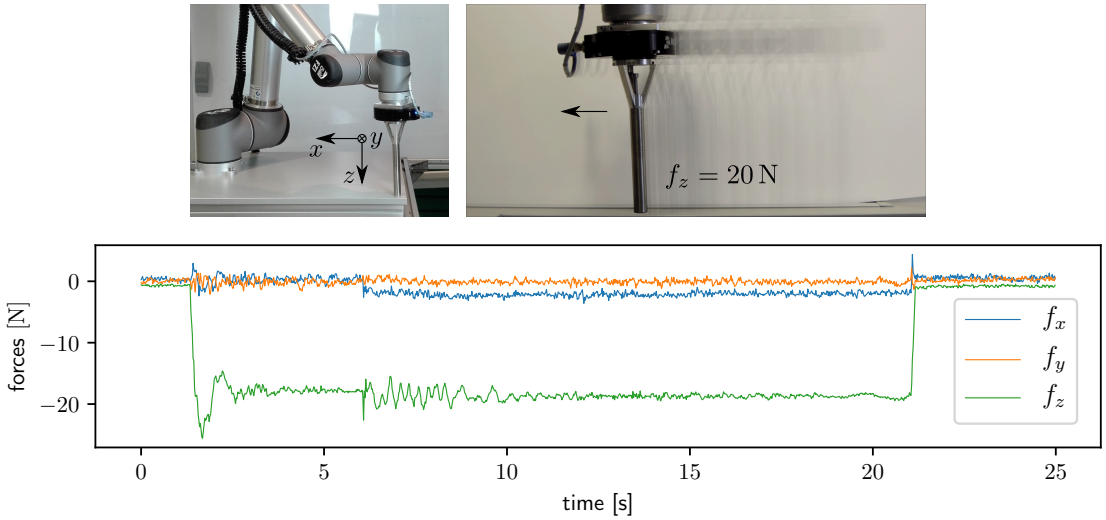


Figure 5.18.: Compliant control in contact. *Top*: Setup of the UR10 with wrist force-torque sensor and steel rod as end-effector. The overlay of several images on the right illustrates the trace of the end-effector during the sliding motion. *Bottom*: Measured contact forces during motion with a nominal target force of $f_z = 20\,\mathrm{N}$. Reproduced and adapted from [216].

5.2.9. Compliant Control

Compliant control allows tracking force profiles during motion. The first experiment evaluates force control in surface-normal direction during sliding in contact. The setup consists of a UR10 robot with a wrist-mounted force-torque sensor, to which we attached the handle from the previous experiment as end-effector. The control rate of our controller on top of the robot's inner position control loop was $125\,\mathrm{Hz}$. In combination with a hard surface of relatively high stiffness, this provides a challenging and thus suitable setup for force control. The motion in parallel is created through a constant stiffness that drives the robot from its start position to a distant target. Fig. 5.18 shows the setup and the measured contact forces. After making the initial contact by setting the force reference, the sliding starts after approximately $6\,\mathrm{s}$. The control law of FDCC proposed a common net force, such that target forces can superimpose virtual stiffness. This effect is visible in the curves in that f_z reaches steady state slightly before the reference of $20\,\mathrm{N}$ due to the target being located few centimeters above the surface. Aligned with classic admittance and impedance control, this joint handling of motion and force subspaces provides more flexibility and avoids updating task-specific selection matrices. Note how the friction of the contact is visible during motion in the measured x-component. After stick-slip phenomena at initiating the sliding motion, force setpoint tracking is smooth.

End-Effector Compliance

The last evaluation of compliance control includes the KR16 robot as a representative of a typical industrial robot for higher payloads. In contrast to the other lightweight systems, its repeatable accuracy of $0.05\,\mathrm{mm}$ targets high-precision tasks. This accuracy is accompanied by very stiff actuators and mechanics, providing a challenging candidate for adding

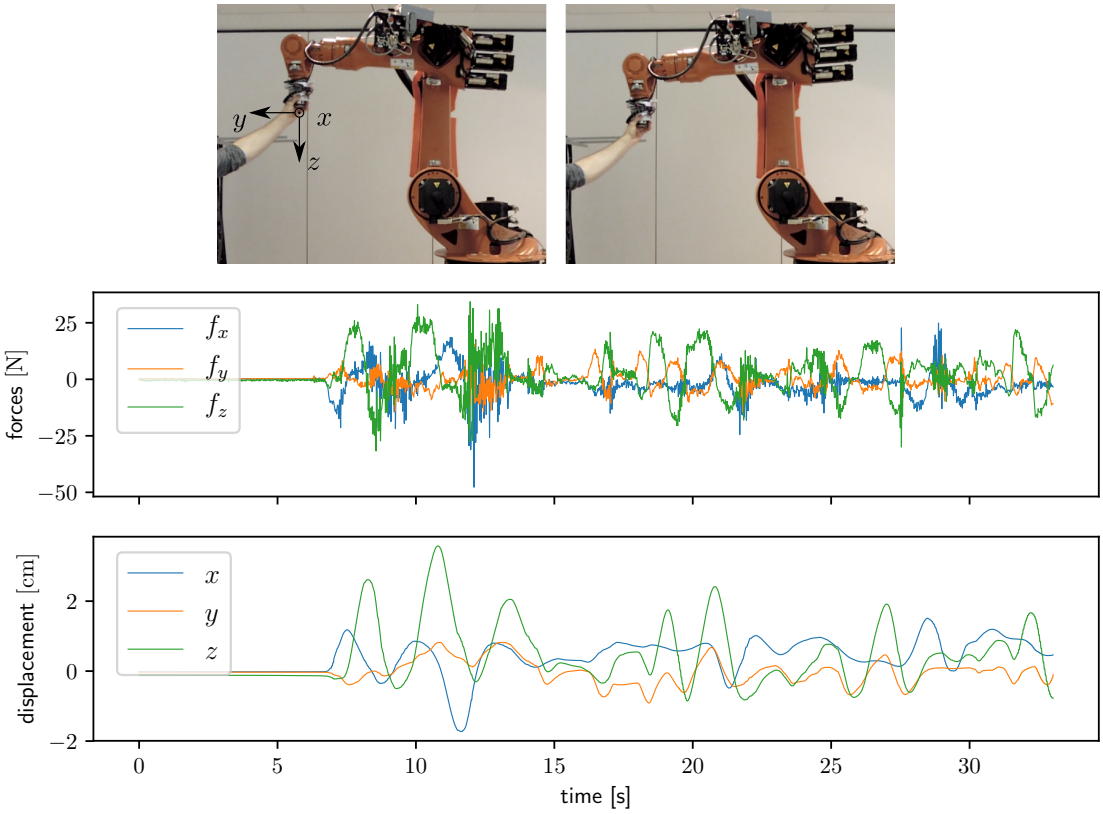


Figure 5.19.: End-effector compliance of the KR16 robot. *Top*: Setting with a wrist-mounted force-torque sensor. *Bottom*: Measured end-effector forces and displacements due to human-robot interaction. The otherwise stiff industrial robot behaves like a spring-damper system. Reproduced and adapted from [216].

end-effector compliance with FDCC. Fig. 5.19 shows the robot with human interaction. As detailed in Section 4.1.2, FDCC superimposes sensor signals to virtual stiffness and damping and forward computes reaction motion. As a result, the manipulator can be parameterized to behave like a spring-damper system at its end-effector. Using the accumulating time integration method from Table 4.3 leads to smooth responses despite unfiltered and noisy measurements.

In a wider context, FDCC was further evaluated during the European Robotics Challenges (EuRoC[5]). In the final use case envisioned, the KR16 robot was used in combination with end-effector compliance control for mounting flexible polymer sealing strips to car doors. Fig. 5.20 shows highlights from the application. Following a reference trajectory around the car door, the robot used search-like patterns in force-controlled contact to set clips of the sealing.

[5]https://cordis.europa.eu/project/id/608849, accessed 04.03.2021

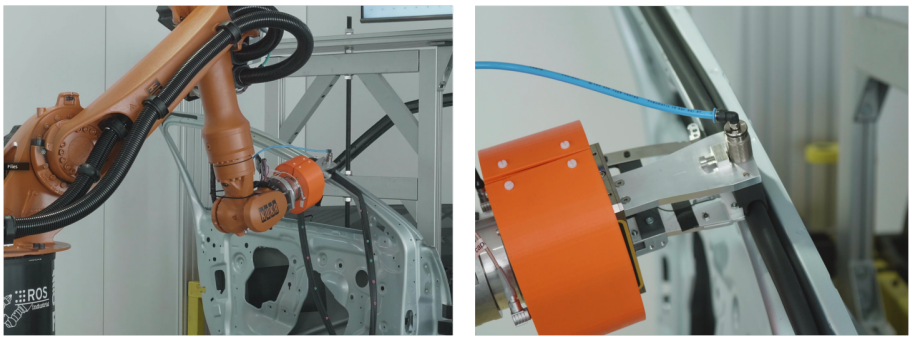


Figure 5.20.: Mounting of flexible polymer sealing strips to car doors with active compliance control. Application of the winning team FLAI^{2}R during the European Robotics Challenges (EuRoC) [220].

5.2.10. Conclusion

The previous sections evaluated the robot controller performance both in simulation and on real robotic manipulators. Experiments on theoretic aspects, in particular on the forward dynamics-based mapping, confirmed task space linearization and robustness in singular joint configurations. This mapping performs similarly to the established DLS method but preserves the inherent stability of the Jacobian transpose. Further experiments tested this approach with FDCC as a controller and showed flexibility for interpolating between waypoints and stability with force control in contact and at workspace limits. The evaluation further showed end-effector compliance to be an easy add-on for industrial robots with wrist-mounted force-torque sensors. Providing these different control interfaces, the proposed FDCC is a suitable basis for skill execution during assembly and manipulator control in general.

5.3. Assembly Use Cases

This last section completes the evaluation of the theory presented in this thesis. Section 5.1 evaluated the skills, their composition, and functioning directly in the simulator and Section 5.2 evaluated various aspects of robot control. We now evaluate the combination of the two components within the skill controllers. For each concrete task, a new assembly skill is created with the primary steps of recording a dataset in simulation and learning this dataset into an instance of the probabilistic forward model. This skill is itself robot independent and formulated in task space coordinates. The controller in contrast is the same for all applications and transforms the skill's predictions into joint control commands of the robot. The first example is a conceptional application within the context of space robotics for on-orbit satellite servicing and the second example solves the toy assembly that served throughout this thesis and shows the transfer to a real robotic system.

5.3.1. Assembly of Satellite Structures

This section evaluates skill controllers on an application for space robotics. The application is part of a wider vision, in which using intelligent Building Blocks for On-Orbit

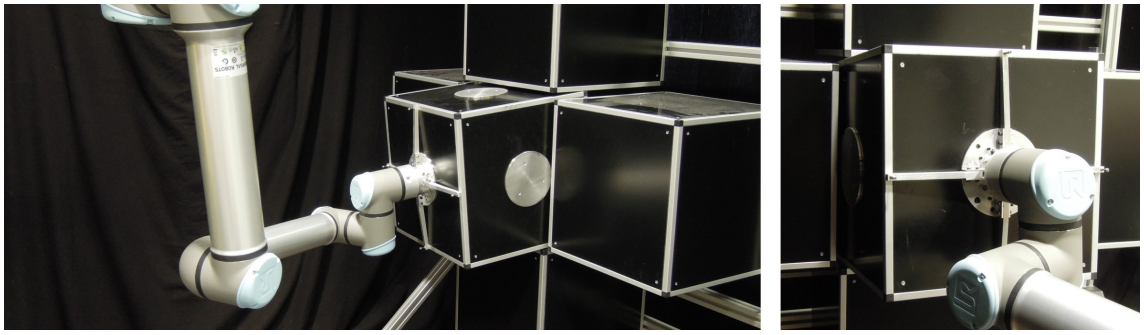


Figure 5.21.: Setup for satellite component assembly with the UR10e robot. The active object is one of five cubes with $400\,\mathrm{mm}$ edge length that is rigidly mounted to the end-effector for insertion.

Satellite Servicing and Assembly (iBOSS) [221] shall modularize and thus economize satellite structures. The key role in this concept take individual components in cube form that are coupled over standardized interfaces to neighboring blocks for data and energy transfer. Each block houses flight electronics and custom payload, and a functioning satellite classically comprises 20 to 30 such cubes. Within this concept, on-orbit maintenance simplifies the exchange of individual cubes, such that a refueling mission, for instance, consists of replacing tank blocks on satellites that adhere to this standard. These missions require an autonomous, robotic execution with space manipulators. Chapter A in the appendix provides additional details on the project and the challenges involved for robotic automation.

Fig. 5.21 shows the simplified setup for the lab demonstrator. It consists of a rack of five blocks, whose middle component is to be inserted with a robotic manipulator. The interfaces for coupling the boxes are approximated through round metal plates of $4\,\mathrm{mm}$ thickness on each face of the cubes. All neighboring blocks possess such plates on their inner sides and they are prone to jam the blocks upon colliding. The clearance of the setup in its final configuration is $4\,\mathrm{mm}$.

Fig. 5.22 shows an execution of the assembly. The control rates were set to $8\,\mathrm{Hz}$ for skill inference and $500\,\mathrm{Hz}$ for robot force control, respectively. The curves of the positions show plateaus on the way to the goal that indicate internal collisions with the side plates. Mere pushing will not lead to success in these spots. Accordingly, the force-torque strategies try various re-orientations to search for clearance and to proceed with the insertion. Note in the f_x component how the skill controller mostly pushes only when progress is made in that direction. We tested the assembly with 100 trials, adding goal uncertainty of varying linear and angular errors of up to $5\,\mathrm{cm}$ and $10\,\mathrm{deg}$, respectively, with an overall success rate of 75 %.

5.3.2. The Toy Assembly

The toy assembly has accompanied the theory of this thesis as an illustrative but challenging example with a multi-axes insertion, low clearance, tight fits, and geometry prone to part jamming. The goal of this last experiment is to test if the skill is portable to real-world systems through our controller. During demonstrations, the simulator allowed partly unrealistic mesh interpenetration to stabilize contact physics, which led to the appearance of higher friction and stiction. This effect was aligned with the concept of making the

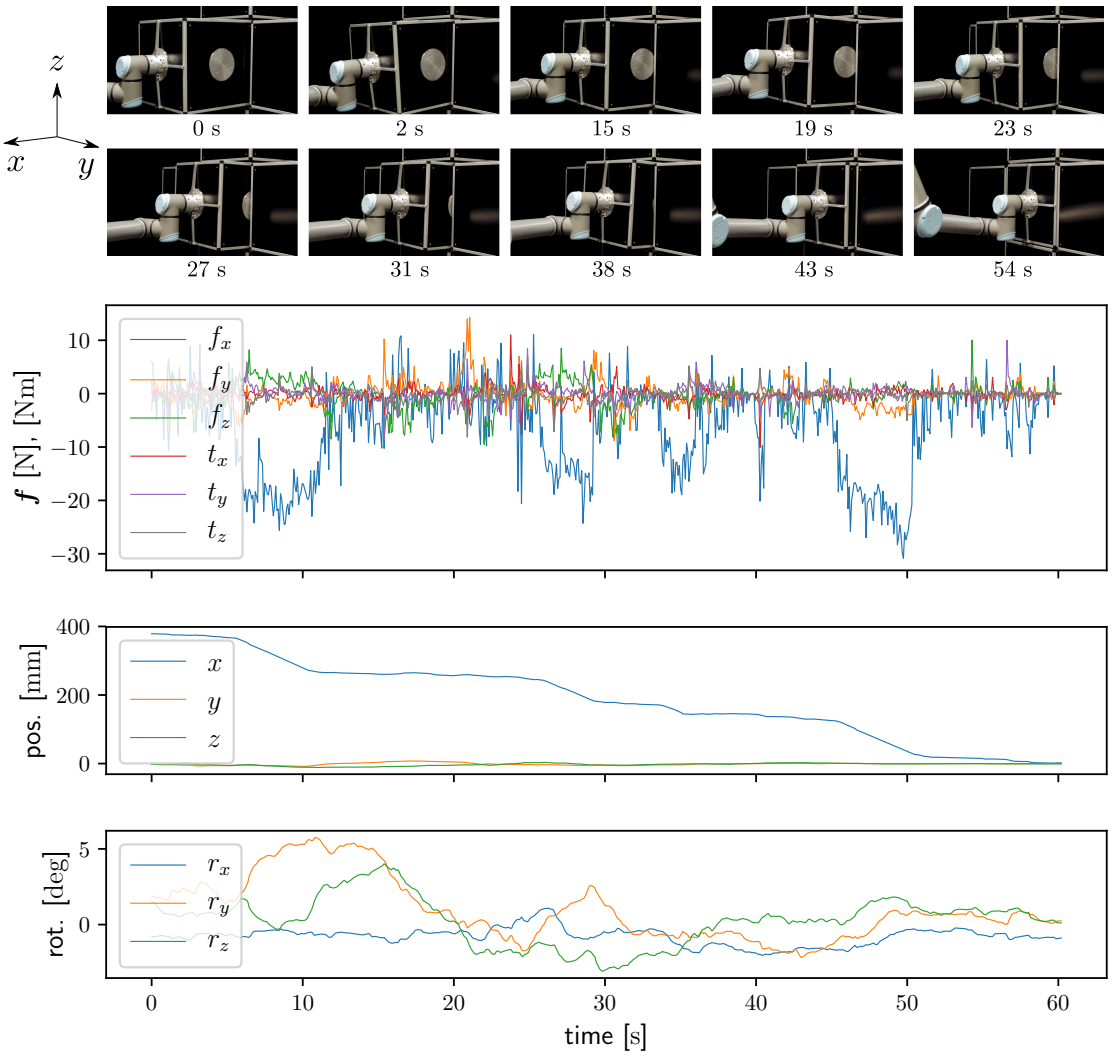


Figure 5.22.: Inserting a satellite module under uncertainty. *Top*: Snapshots of the assembly with internal collisions during the execution. *Bottom*: The force-torque strategies as commanded by the skill model and the cube's relative pose during robot control.

Figure 5.23.: Setup for the toy assembly with the UR10e robot. *Left*: The passive object is clamped in a vice and glued to a surface with adhesive tape. *Right*: A non-actuated gripper holds the active object in place with glued jaws. The elasticity of the material adds uncertainty to the system.

assembly more difficult in simulation to favor an easier transfer to real hardware. While friction is lower on the real setup for this combination of plastic components, the low clearance and tight fits, especially at the final stage of the assembly, are similar.

Fig. 5.23 shows the setup. We chose a passive gripper for this experiment that holds the active assembly object in place relative to the robot end-effector. We obtained an approximate goal pose by teleoperating the robot through the assembly once in free force control with the teach device and recording the final configuration. During execution, the skill model's inference was set at $5\,\mathrm{Hz}$, yielding $200\,\mathrm{ms}$ for each setpoint of the predicted force-torque reference profile. FDCC was running in free force control on the robot at $500\,\mathrm{Hz}$ and controlling the UR10e robot over its joint position control interface. Fig. 5.24 shows an execution with intermediate part jamming that the skill controller needed to solve. After establishing contact, the parts quickly jam, and strategies are needed to advance. Note how the controller makes various push attempts into the principal forward direction, as is visible in the negative swings of the f_x component. Since this is not immediately successful, different re-orientations follow to search for part clearance. After searching approximately $90\,\mathrm{s}$ in this area, the skill finds the entrance and plugs both parts together.

Performance is usually decreased for configurations that are strongly different from those contained in the dataset. There are two local optima, where parts can be plugged in intermediate configurations, such that the controller struggled and partially failed in these cases. Once these shortcomings are known, the preparation of more specific demonstrations for these error cases can increase robustness for practical applications.

The experience from the task allowed us to chose suitable ranges for forces and torques previously in simulation so that the skill was trained with a nominal *intensity*. Without this task knowledge at hand, some form of calibration will be necessary to scale the skill predictions to suitable ranges. This adaptation is, however, easy due to forces and torques being vector quantities, and the reference profile can be scaled globally or individually per axis during execution.

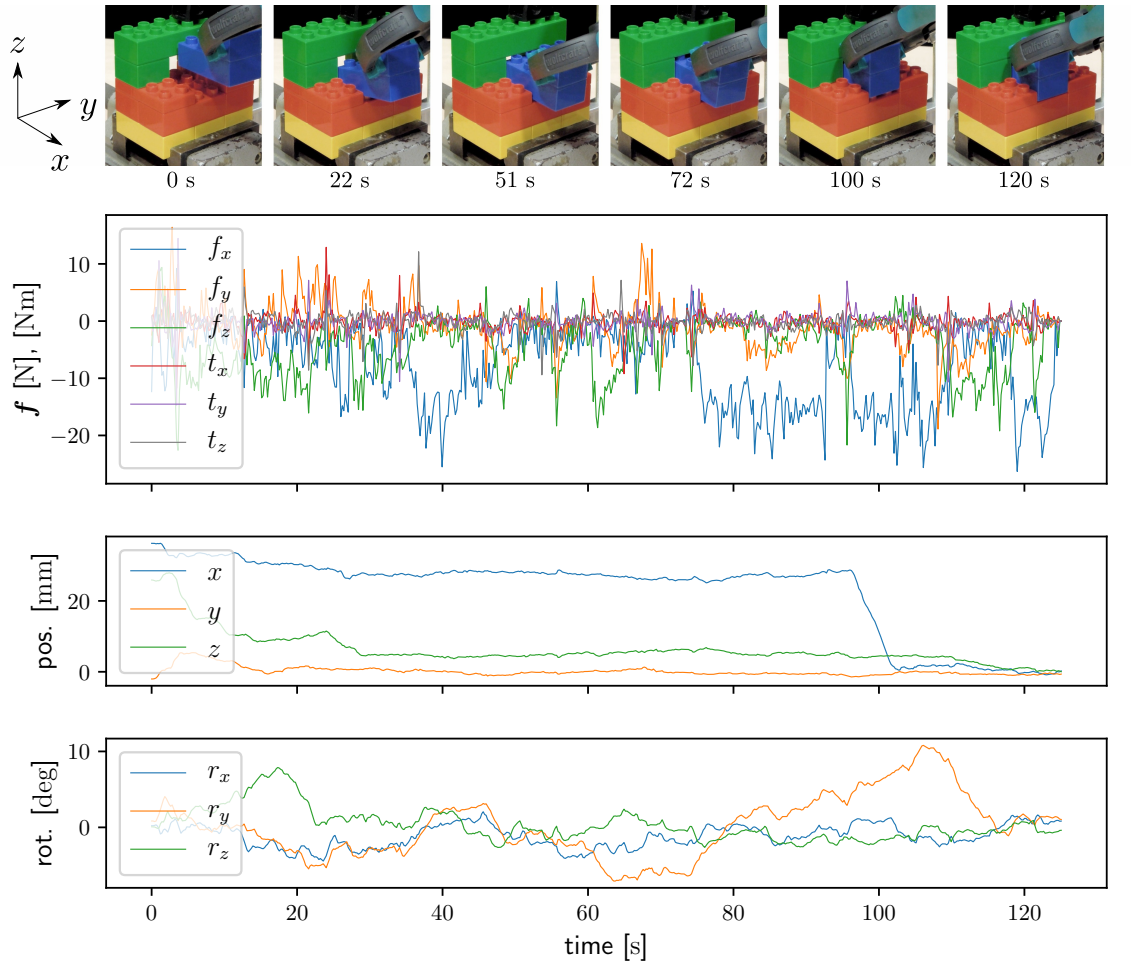


Figure 5.24.: Successful toy assembly with intermediate part jamming due to uncertainty. *Top*: An excerpt of recorded snapshots during the execution. *Bottom*: The skill's inferred force-torque commands for robot control and the goal-relative pose of the grasped object during progress.

5.3.3. Conclusion

Skill controllers provide an effective means to transfer and execute assembly skills that are trained on demonstrations in simulation. The two applications required each skill to have its dataset to learn from. A possible transfer of skills between tasks was beyond the scope of this thesis.

Maintaining contact stability in force control currently demands comparatively low execution speeds. This limitation is characteristic of motion-actuated systems and inherent in admittance control with industrial robots in rigid contacts. Low clearances and highly constrained objects during assembly emphasize this necessity in general, which in turn means to slow down the skill inference rate to create similar conditions as was used during demonstration. This behavior was anticipated and supported by the skill controllers' flexibility to scale the force-torque strategies both in time and in intensity.

6. Summary

The programming of robotic assembly tasks is an important component of manufacturing. Industrial robots are common-place for fast, repetitive, and unergonomic motion but less established for contact-rich tasks, where the manual baseline is still hard to come by.

This thesis focussed on three associated problem areas: assembling tight-fitting parts with low clearance, handling uncertainty and offsets in positioning, and providing a suitable method to offline program both for industrial settings. The intersection of the three areas constitutes a major obstacle for automation, and each of these areas has its own challenges, which are hard to overcome. Low clearances can lead to jamming and wedging of workpieces, in such a way that an open-loop trajectory execution is not feasible and even slight offsets must be compensated online with reactive behavior. Although strategies are applied intuitively by human workers, they are not easy to describe analytically for robot programs. And achieving this offline adds even further limitations.

This thesis proposed teleoperation in simulation as a form of PbD and a technical representation of assembly skills with temporal-probabilistic neural networks. The work derived an intuitive programming method to transfer human-inspired capabilities into robot controllers, for which Chapter 3 and Chapter 4 developed the theory, and Chapter 5 provided the experimental evaluation.

The following Section 6.1 summarizes important insights in the light of the initial research questions, Section 6.2 then discusses limitations and points to future research, and Section 6.3 concludes with the contributions of this thesis and provides an outlook.

6.1. Discussion

Q-I *How to design a simple and intuitive method to program robotic skills for force-sensitive assembly tasks offline?*

Offline programming poses the challenge of an inaccessible robot and application during programming. The PbD paradigm is intuitive and can be realized by combining teleoperation with simulation. A simple joystick-like teach-device is sufficient to show the assembly by steering the active object in all six Cartesian dimensions and thus making use of the implicitly available experience in object manipulation. The two important challenges of numerical instability and the lack of precisely known process parameters can be addressed by tolerating unrealistic mesh-interpenetration and artificially increasing friction. The environments are designed to motivate reactive and robust strategies during the demonstration that can compensate for this degraded realism on the real setups.

Q-II *How to extract and incorporate human skill and intuition to make robots handle part jamming and wedging autonomously during execution?*

This thesis followed a data-driven approach and based skills on the idea to mimic mental forward models, in which the differences between expected and observed

state transitions trigger intuitive strategies for correction. By provoking challenging part constellations with jamming in the simulator, it becomes necessary to free the parts from intermediate deadlocks during the demonstration. It is possible to solve the tasks without force feedback by observing the outcome of our own actions in the simulator. A technical representation of this task-specific skill can be achieved by learning a policy from these recordings. Suitable models must be able to capture patterns in ambiguous strategies for tight-fitting workpieces, which is achieved by the proposed temporal-probabilistic neural networks. During execution, the interplay between initiated motor command and response is concise enough to solve the tasks without further visual perception of the robotic systems.

Q-III *How to implement controllers to transfer these offline-programmed skills to industrial manipulators?*

A separation of skill and robot control allows for a flexible deployment within the manipulator's workspace and requires a force controller that operates within the Cartesian regime. The skill transfer can be achieved by incorporating forward dynamics algorithms into robot control and thereby mimicking the simulated behavior during demonstrations. This leads to a new control paradigm and a suitable alternative to current IK approaches for admittance control in general. The new controller is applicable for joint position and joint velocity interfaces that are widely available on industrial robots.

Robotic assembly is a rich and diverse field that spans several decades of research. Chapter 2 reviewed existing approaches and derived important criteria to identify strengths and shortcomings in Section 2.3. This thesis proposed the combination of the skill-based approach with the PbD paradigm over Cartesian manipulator control. We now briefly discuss our contribution under the initial criteria:

(1) Compatibility with motion-actuated systems The industrial robots considered are non-backdrivable in their joints and rely on active compliance by control. FDCC is tailored for these systems and implements suitable interfaces for force-motion setpoints within the Cartesian regime. When combined with assembly skills, the controller transforms force-torque strategies and measurements from wrist-mounted sensors directly into reference joint motion on the robots.

(2) Suitability for low clearances and tight fits We approximate the contacts between tightly fitting parts with spring-damper elements. This leads to partially unrealistic physics during demonstrations, but the recordings are concise enough for learning assembly skills for the real setups. This makes the proposed approach applicable to a wide range of challenging assembly tasks such as plug-insertions.

(3) Suitability for arbitrary masses/dimensions The proposed method is conceptionally independent of workpiece dimensions. Masses and inertia parameters are arbitrary during demonstrations and the focus is on designing a velocity-proportional response to force-torque commands both during simulation and control.

(4) Suitability for simulation and transfer The concept builds on simulation both during skill demonstration and robot control. But instead of aiming for highly accurate models and process parameters, it supports the learning of robot-independent skills in these imperfect environments. The transfer to real robotic manipulators is achieved with FDCC that mimics the simulator during execution.

(5) Handling of uncertainty/jamming/wedging This aspect was included by provoking challenging configurations in simulation and learning from human behavior. The trained skill models can compensate for uncertain goal poses to some extent.

(6) Intuitiveness of programming Teleoperation is not as intuitive as showing the assembly directly with our own hands, and using the teach device in the simulator must be learned. With growing experience, however, neither background in robotics nor engineering is required to program assembly skills with this method.

(7) Independence of the task complexity Our concept is based on using mesh data from CAD and the process of simulation and demonstration is the same for simple and complex workpieces. However, the solver's collision checking and contact simulation may limit the simulator's real-time performance. If this becomes a bottleneck, the simulation may be slowed down accordingly during demonstrations. An alternative is to reduce the triangle count of the meshes before setting up the simulator.

6.2. Limitations and Further Research

Chapter 5 evaluated the research with experiments and showed the validity of the proposed methods. The following sections discuss limitations and indicate promising next steps.

6.2.1. Skill Generalization and Fine Tuning

Our concept so far produces task-specific skills. They are demonstrated and learned for a combination of two workpieces. While this is suitable for special components whose assembly is difficult to program otherwise, there are limitations for assemblies that require skills for hundreds of components. Using simulation partially alleviates this problem by enabling more people to demonstrate assembly tasks without requiring actual hardware. A distributed, crowd-based skill collection could be a promising path [152]. Having more data available on a variety of different assemblies also motivates to explore the generality of these strategies. Supervised Learning can provide powerful models to achieve such a generalization if sufficient data are available. Skill models with massive parameters might learn general core strategies in huge datasets that can be deployed for new tasks. A suitable approach could then be to use these pre-trained models and fine-tune them with few demonstrations on the actual use cases [222].

6.2.2. Elastic Workpieces

Rigid objects formed the baseline assumption about assembly processes throughout this thesis. An interesting path for future research is to explore our approach for partly elas-

tic workpieces. With the assumption that elastic deformation is somehow characteristic, learning part deformation could be achieved implicitly by learning non-linear motion response to force-torque commands. This would include the material stiffness of the components in jamming scenarios, similar to a geometry-dependent, 6-dimensional spring that the skills need to learn.

While the concept itself is transparent for both rigid and elastic approximations in simulation, the latter is more difficult to model. Fewer simulation environments do support this functionality by default. In addition, collision checking becomes more evolved and computational performance will currently limit a fast and intuitive demonstration by slowing-down the simulator. A slight remedy is that our concept does not require high realism, and course approximations of simulating elastic components could be suitable for skill extraction.

6.2.3. Robot Control

We focused on motion-actuated robots in this thesis. They are widely in use and currently without alternatives especially for the handling of heavy components. Providing active compliance by control, however, directly depends on the control rate of their inner joint control loops, limiting the FDCC algorithm to relatively slow motion in contact with stiff environments. At least for small and for medium-sized parts, lightweight robots with advanced sensors in the wrist, joints, and basis [11] provide powerful capabilities for fast and sensitive interaction. It would be interesting to test the limits of the skill execution on these platforms.

6.3. Conclusion and Outlook

This thesis followed the incentive to bring human-inspired skills to compliant robot control and made the following contributions:

- A literature survey on skill programming and skill learning in robotics research with a focus on robotic assembly. This collection highlighted characteristics and qualitatively evaluated strengths and weaknesses.
- An approach to capture human-inspired assembly skills with teleoperation in simulation and to model these skills with sequential-probabilistic neural networks. Used as a cognitive overlay, robots can deploy these error-correcting strategies against unforeseen part jamming.
- A unifying compliant controller tailored for industrial manipulators to execute these assembly skills. The controller builds on simplified dynamics simulations and provides a new control concept and solution to the inverse kinematics problem for the field of manipulator control.

The skillful handling of compliant interactions is at the heart of robotic assembly and a decisive step towards higher grades of autonomy and automation. Exciting challenges lie ahead towards robots that operate with human-level skills in these settings. The sensor processing and actuation determine how the robots interact with their environments and which complexity they can technically master. For the assemblies considered, the

robots would typically have special end-effectors. As long as the robots can apply the strategies through compliant controllers, more complex gripper morphologies could be used similarly.

In this direction, anthropomorphic robot hands will provide more degrees of freedom and support assembly skills through additional dexterity. Likewise, perception, scene understanding, and high-level reasoning are required to update the simulated models online and select the skills appropriately. The rich experience of humans in object manipulation is currently unmatched but expanding the skill input to all relevant human senses and learning from enormous amounts of data will eventually lead to general assembly strategies that come truly close to those of human workers.

A. On-Orbit Satellite Assembly

Vision and Application

Numerous key technologies of modern societies depend directly or indirectly on satellites. Their reliability is determined by deterministic and non-deterministic events, such as running out of fuel or damage through space debris. The on-orbit servicing of spacecraft could effectively mitigate frequent failures [223] but requires a rethinking of the classic, monolithic design. Fig. A.1 illustrates this concept: A special servicer satellite is equipped with robotic manipulators for the maintenance and assembly of a client satellite. The intelligent Building Blocks for On-Orbit Satellite Servicing and Assembly (iBOSS) [221] envision a fully modular approach. Standardized, box-shaped components shall economize these spacecraft for the private sector. The components carry flight electronics and custom payload and are combined in a three-dimensional grid to function as a distributed system. Fig. A.2 shows the smallest of such building blocks. They are coupled over the intelligent Space System Interface (iSSI) on their sides, which enables the transfer of mechanical load, power, heat, and data between neighboring blocks.

The usage of standardized components simplifies the robotic assembly of satellite structures and their maintenance to the arrangement and replacement of individual blocks. Once the components are brought into position, a motor-actuated bayonet clutch closes the connection for each interface. This mechanism comprises both male and female functionality and secures a redundancy against failure. As a consequence, the servicer satellites can manipulate a non-responding and non-cooperating client satellite, such as during re-fueling missions.

Challenges for Robotic Manipulation

Similar to industrial applications, the challenge lies within the automation of human skills during task execution. Human-teleoperated maintenance and assembly through operators at a ground station are ruled out through insufficient uplink and downlink data rates for visual or haptic feedback. The uncertainty through localization and the limited accuracy of current robotic manipulators in space require reactive strategies to handle colliding iSSIs during the insertion of neighboring blocks. These strategies need to search carefully for part clearance during insertion to get the components into place. After programming these skills offline, the servicer satellite needs to execute the tasks autonomously and robustly with its robotic manipulators.

[1] Image source:
http://www.iboss-satellites.com/fileadmin/Templates/iBOSS_Satellites/Media/iBOSS_Concept.pdf
accessed 27.3.2021

[2] Image source:
http://www.iboss-satellites.com/fileadmin/Templates/iBOSS_Satellites/Media/iBLOCK.pdf
accessed 27.3.2021

Figure A.1.: Schematic rendering of an on-orbit servicing scenario in the iBOSS concept[1].

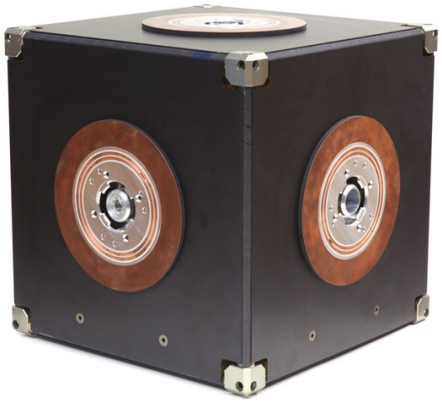


Figure A.2.: The intelligent Building Block (iBLOCK)[2].

B. Software Implementation

The software-side contribution of this thesis builds upon various components that are developed in different frameworks. Table B.1 shows an overview of the most important ones. General program logic and additional scripts for the experiments are not displayed for clarity. The footnotes point to central libraries within the respective framework that we used for each component.

The column for contributions has the following meaning: Some components are mainly *used* and already provide suitable functionality and mechanisms to realize our ideas. They were parameterized accordingly to be compatible with our requirements. All of them are free and openly available. Other components needed more customization and development to connect them to our overall architecture, such as plugins and interfacing scripts. They are referred to with *interfaced*. The *implemented* components were fully developed in this thesis from the presented theory. FDCC was published in form of a free and open source software package *cartesian_controllers*[7]. More components are planned to be published in the future.

[1]http://gazebosim.org/ , accessed 27.3.2021
[2]https://www.blender.org/ , accessed 27.3.2021
[3]https://www.ode.org/ , accessed 7.8.2020
[4]https://github.com/ros-drivers/joystick_drivers/tree/main/spacenav_node
[5]https://github.com/tensorflow/tensorflow/blob/r2.0/tensorflow/python/keras/layers/recurrent.py
[6]https://www.tensorflow.org/versions/r2.1/api_docs/python/tf , accessed 27.3.2021
[7]https://github.com/fzi-forschungszentrum-informatik/cartesian_controllers
[8]https://github.com/ros-industrial/ur_modern_driver
[9]https://github.com/UniversalRobots/Universal_Robots_ROS_Driver
[10]https://github.com/ipa320/schunk_modular_robotics
[11]https://github.com/ros-industrial/kuka_experimental

Table B.1.: Software components of this thesis.

Component	Theory	Framework	Contribution	Language
Simulator	Section 3.2.1	Gazebo[1]	*interfaced*	-
Meshes	Fig. 3.2	Blender[2]	*implemented*	-
Contact physics	Section 3.2.2	ODE[3]	*used*	-
Teach device	Section 3.2.3	ROS[4]	*interfaced*	Python
Recording	Section 3.2.3	ROS	*implemented*	Python
LSTM layer	Fig. 3.13	Tensorflow[5]	*used*	Python
MDN layer	Fig. 3.14	Tensorflow	*implemented*	Python
Skill model	Fig. 3.15	Tensorflow	*implemented*	Python
Data composition	Fig. 3.16	ROS	*implemented*	Python
Training	Section 3.3.4	Tensorflow[6]	*used*	Python
FDCC	Section 4.3.2	ROS-control	*implemented*[7]	C++
Skill controller	Fig. 4.7	ROS	*implemented*	C++, Python
UR10	-	driver[8,9]	*used*	C++
LWA4P	-	driver[10]	*used*	C++
KR16	-	driver[11]	*used*	C++

C. Prior Publications

Journal Preprints

(1) S. Scherzinger, A. Roennau, and R. Dillmann. "Virtual Forward Dynamics Models for Cartesian Robot Control" In *Journal of Intelligent & Robotic Systems* (under review) Preprint available at arXiv 2009.11888 [cs.RO], Sep. 2020

Book Chapters

(1) G. Heppner, F. Mauch, S. Scherzinger, et al. "FLA2IR—FLexible Automotive Assembly with Industrial Co-workers" In *Bringing Innovative Robotic Technologies from Research Labs to Industrial End-users: The Experience of the European Robotics Challenges* Ed. by Fabrizio Caccavale et al. Cham: Springer International Publishing, pages 97–126, 2020

Conferences

(1) S. Scherzinger, A. Roennau, and R. Dillmann. "Inverse Kinematics with Forward Dynamics Solvers for Sampled Motion Tracking" In *19th International Conference on Advanced Robotics (ICAR)*, pages 681–687, Dec. 2019.

(2) S. Scherzinger, A. Roennau, and R. Dillmann. "Contact Skill Imitation Learning for Robot Independent Assembly Programming" In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4309–4316, Nov. 2019.

(3) S. Scherzinger, A. Roennau, and R. Dillmann. "Forward dynamics compliance control (fdcc): A new approach to cartesian compliance for robotic manipulators". In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4568–4575, Sep. 2017.

Workshops

(1) S. Scherzinger "cartesian_controllers: Motion, Force, and Compliance Control for Robotic Manipulators" *ROSCon 2019*, Macau, Nov. 2019, Slides available at https://roscon.ros.org/2019/talks/roscon2019_cartesiancontrollers.pdf
Talk available at https://vimeo.com/378682968

(2) S. Scherzinger, S. Rühl, J. Mangler, A. Roennau, and R. Dillmann. "Robotic Manipulation for On-Orbit Satellite Servicing – Intermediate Results from the iBOSS Project" Workshop on *The Next Generation of Space Robotic Servicing Technologies* At *2015 IEEE International Conference on Robotics and Automation (ICRA)* , May 2015.

Bibliography

[1] Daniel E Whitney. "Historical perspective and state of the art in robot force control". In: *The International Journal of Robotics Research* 6.1 (1987), pp. 3–14.

[2] H. Asada. "Teaching and learning of compliance using neural nets: representation and generation of nonlinear compliance". In: *Proceedings., IEEE International Conference on Robotics and Automation*. 1990, 1237–1244 vol.2.

[3] Zengxi Pan et al. "Recent progress on programming methods for industrial robots". In: *Robotics and Computer-Integrated Manufacturing* 28.2 (2012), pp. 87–94.

[4] S. Gottschlich, C. Ramos, and D. Lyons. "Assembly and task planning: a taxonomy". In: *IEEE Robotics Automation Magazine* 1.3 (1994), pp. 4–12.

[5] Rüdiger Dillmann, M Kaiser, and Ales Ude. "Acquisition of elementary robot skills from human demonstration". In: *International Symposium on Intelligent Robotics Systems*. Citeseer. 1995, pp. 185–192.

[6] Bram Vanderborght et al. "Variable impedance actuators: A review". In: *Robotics and autonomous systems* 61.12 (2013), pp. 1601–1614.

[7] Daniel E Whitney. "Quasi-static assembly of compliantly supported rigid parts". In: *Journal of Dynamic Systems, Measurement, and Control* 104.1 (1982), pp. 65–77.

[8] M. T. Mason. "Compliance and Force Control for Computer Controlled Manipulators". In: *IEEE Transactions on Systems, Man, and Cybernetics* 11.6 (June 1981), pp. 418–432.

[9] Roland Carlyle Groome. "Force feedback steering of a teleoperator system." PhD thesis. Massachusetts Institute of Technology, 1972.

[10] James L Nevins and Daniel E Whitney. "The force vector assembler concept". In: *On Theory and Practice of Robots and Manipulators*. Springer, 1972, pp. 273–288.

[11] M. Iskandar et al. "Joint-Level Control of the DLR Lightweight Robot SARA". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 8903–8910.

[12] Rainer Bischoff et al. "The KUKA-DLR Lightweight Robot arm-a new reference platform for robotics research and manufacturing". In: *Robotics (ISR), 2010 41st international symposium on and 2010 6th German conference on robotics (ROBOTIK)*. VDE. 2010, pp. 1–8.

[13] Alin Albu-Schaffer and Gerd Hirzinger. "Cartesian impedance control techniques for torque controlled light-weight robots". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. 2002, pp. 657–663.

[14] D. Surdilovic. "Contact stability issues in position based impedance control: theory and experiments". In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 2. Apr. 1996, 1675–1680 vol.2.

[15] F. Lange, M. Suppa, and G. Hirzinger. "Control with a Compliant Force-Torque Sensor". In: *ROBOTIK 2012; 7th German Conference on Robotics*. 2012, pp. 1–6.

[16] F. Lange, W. Bertleff, and M. Suppa. "Force and trajectory control of industrial robots in stiff contact". In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 2927–2934.

[17] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. 2009, p. 5.

[18] K. G. Shin and C. Lee. "Compliant control of robotic manipulators with resolved acceleration". In: *1985 24th IEEE Conference on Decision and Control*. 1985, pp. 350–357.

[19] J. Luh, M. Walker, and R. Paul. "Resolved-acceleration control of mechanical manipulators". In: *IEEE Transactions on Automatic Control* 25.3 (June 1980), pp. 468–474.

[20] Marc H Raibert and John J Craig. "Hybrid position/force control of manipulators". In: *Journal of Dynamic Systems, Measurement, and Control* 102.127 (1981), pp. 126–133.

[21] Chi-Haur Wu and Richard P Paul. "Resolved motion force control of robot manipulator". In: *IEEE Transactions on Systems, Man, and Cybernetics* 12.3 (1982), pp. 266–275.

[22] O. Khatib. "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *IEEE Journal on Robotics and Automation* 3.1 (Feb. 1987), pp. 43–53.

[23] Oussama Khatib. "Inertial Properties in Robotic Manipulation: An Object-Level Framework". In: *The International Journal of Robotics Research* 14.1 (1995), pp. 19–36.

[24] Luigi Villani and Joris De Schutter. "Force control". In: *Springer handbook of robotics* (2008), pp. 161–185.

[25] A. Calanca, R. Muradore, and P. Fiorini. "A Review of Algorithms for Compliant Control of Stiff and Fixed-Compliance Robots". In: *IEEE/ASME Transactions on Mechatronics* 21.2 (2016), pp. 613–624.

[26] Marie Schumacher et al. "An introductory review of active compliant control". In: *Robotics and Autonomous Systems* 119 (2019), pp. 185–200.

[27] J.J. Craig and M. Raibert. "A systematic method of hybrid position/force control of a manipulator". In: *Computer Software and Applications Conference, 1979. Proceedings. COMPSAC 79. The IEEE Computer Society's Third International*. 1979, pp. 446–451.

[28] Joris De Schutter and Hendrik Van Brussel. "Compliant robot motion II. A control approach based on external control loops". In: *The International Journal of Robotics Research* 7.4 (1988), pp. 18–33.

[29] S. Chiaverini and L. Sciavicco. "The parallel approach to force/position control of robotic manipulators". In: *IEEE Transactions on Robotics and Automation* 9.4 (1993), pp. 361–373.

[30] Bruno Siciliano. "Parallel Force/Position Control of Robot Manipulators". In: *Robotics Research*. Ed. by Georges Giralt and Gerhard Hirzinger. London: Springer London, 1996, pp. 78–89.

[31] Homayoun Seraji. "Adaptive admittance control: An approach to explicit force control in compliant motion". In: *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE. 1994, pp. 2705–2712.

[32] N. Hogan. "Impedance control - An approach to manipulation. I - Theory. II - Implementation. III - Applications". In: *ASME Transactions Journal of Dynamic Systems and Measurement Control B* 107 (Mar. 1985), pp. 1–24.

[33] Christian Ott, Ranjan Mukherjee, and Yoshihiko Nakamura. "Unified impedance and admittance control". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2010, pp. 554–561.

[34] J. K. Salisbury. "Active stiffness control of a manipulator in cartesian coordinates". In: *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*. Dec. 1980, pp. 95–100.

[35] Daniel E Whitney. "Force feedback control of manipulator fine motions". In: (1977).

[36] Serdar KuCuk and Zafer Bingul. "The inverse kinematics solutions of industrial robot manipulators". In: *Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM'04.* IEEE. 2004, pp. 274–279.

[37] Rosen Diankov et al. "Manipulation planning for the jsk kitchen assistant robot using openrave". In: *The 29th Annual Conference on Robotics Society of Japan, AC2Q2–2*. 2011.

[38] Patrick Beeson and Barrett Ames. "TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea, Nov. 2015.

[39] Samuel R Buss. "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods". In: *IEEE Journal of Robotics and Automation* 17.1-19 (2004), p. 16.

[40] Tsuneo Yoshikawa. "Manipulability of Robotic Mechanisms". In: *The International Journal of Robotics Research* 4.2 (1985), pp. 3–9.

[41] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[42] Yoshihiko Nakamura and Hideo Hanafusa. "Inverse kinematic solutions with singularity robustness for robot manipulator control". In: *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control* 108 (1986), pp. 163–171.

[43] C. W. Wampler. "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods". In: *IEEE Transactions on Systems, Man, and Cybernetics* 16.1 (Jan. 1986), pp. 93–101.

[44] Samuel R Buss and Jin-Su Kim. "Selectively damped least squares for inverse kinematics". In: *Journal of Graphics tools* 10.3 (2005), pp. 37–49.

[45] Marc G Carmichael, Dikai Liu, and Kenneth J Waldron. "A framework for singularity-robust manipulator control during physical human-robot interaction". In: *The International Journal of Robotics Research* 36.57 (2017), pp. 861–876.

[46] J. M. Schimmels and M. A. Peshkin. "Admittance matrix design for force-guided assembly". In: *IEEE Transactions on Robotics and Automation* 8.2 (1992), pp. 213–227.

[47] W.S. Newman et al. "Force-responsive robotic assembly of transmission components". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3. 1999, 2096–2102 vol.3.

[48] J. M. Schimmels and M. A. Peshkin. "The space of admittance control laws that guarantees force-assembly with friction". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1993, pp. 443–448.

[49] Shuguang Huang and J. M. Schimmels. "Sufficient conditions used in admittance selection for force-guided assembly of polygonal parts". In: *IEEE Transactions on Robotics and Automation* 19.4 (2003), pp. 737–742.

[50] S. Huang and J. M. Schimmels. "Design of Spatial Admittance for Force-Guided Assembly of Polyhedral Parts in Single Point Frictional Contact". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5801–5807.

[51] X. Ji and J. Xiao. "Automatic generation of high-level contact state space". In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 1. 1999, pp. 238–244.

[52] Jing Xiao and Xuerong Ji. "Automatic generation of high-level contact state space". In: *The International Journal of Robotics Research* 20.7 (2001), pp. 584–606.

[53] Peng Tang and Jing Xiao. "Automatic generation of high-level contact state space between 3d curved objects". In: *The International Journal of Robotics Research* 27.7 (2008), pp. 832–854.

[54] W. Meeussen et al. "Integration of planning and execution in force controlled compliant motion". In: *Robotics and Autonomous Systems* (2008), pp. 437–450.

[55] A. Stemmer, A. Albu-Schaffer, and G. Hirzinger. "An Analytical Method for the Planning of Robust Assembly Tasks of Complex Shaped Planar Parts". In: *Robotics and Automation, 2007 IEEE International Conference on*. Apr. 2007, pp. 317–323.

[56] A. Stemmer et al. "Robust Assembly of Complex Shaped Planar Parts Using Vision and Force". In: *2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. 2006, pp. 493–500.

[57] M. Suomalainen and V. Kyrki. "Learning compliant assembly motions from demonstration". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016, pp. 871–876.

[58] David R. Strip. *Technology for robotic mechanical assembly: Force-directed insertions*. AT&T Technical Journal. 1988.

[59] H. Onda, H. Hirukawa, and K. Takase. "Assembly motion teaching system using position/force simulator - extracting a sequence of contact state transition". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1995, pp. 9–16.

[60] H. Onda et al. "Assembly motion teaching system using position/force simulator-generating control program". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1997, pp. 938–945.

[61] T. Hasegawa, T. Suehiro, and K. Takase. “A model-based manipulation system with skill-based execution”. In: *IEEE Transactions on Robotics and Automation* 8.5 (Oct. 1992), pp. 535–544.

[62] J. D. Morrow and P. K. Khosla. “Sensorimotor primitives for robotic assembly skills”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 1995, pp. 1894–1899.

[63] J. D. Morrow and P. K. Khosla. “Manipulation task primitives for composing robot skills”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. 1997, pp. 3354–3359.

[64] H. Mosemann and F. M. Wahl. “Automatic decomposition of planned assembly sequences into skill primitives”. In: *IEEE Transactions on Robotics and Automation* 17.5 (2001), pp. 709–718.

[65] U. Thomas et al. “Error-tolerant execution of complex robot tasks based on skill primitives”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 3. 2003, 3069–3075 vol.3.

[66] K. Nottensteiner et al. “A Complete Automated Chain for Flexible Assembly using Recognition, Planning and Sensor-Based Execution”. In: *Proceedings of ISR 2016: 47st International Symposium on Robotics*. 2016, pp. 1–8.

[67] A. Wahrburg et al. “Combined pose-wrench and state machine representation for modeling Robotic Assembly Skills”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 852–857.

[68] U. Thomas et al. “A new skill based robot programming language using UML/P Statecharts”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 461–466.

[69] Arvid Butting et al. “Modeling reusable, platform-independent robot assembly processes”. In: *arXiv preprint arXiv:1601.02452* (2016).

[70] H. Fakhurldeen, F. Dailami, and A. G. Pipe. “CARA system Architecture - A Click and Assemble Robotic Assembly System”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5830–5836.

[71] R. H. Taylor T. Lozano-Perez M. T. Mason. “Automatic Synthesis of Fine-Motion Strategies for Robots”. In: *Int. J. Robotics. Res.* Vol. 3.No. 1 (Dec. 1984), pp. 3–24.

[72] J. De Schutter and H. Van Brussel. “Compliant Robot Motion I. A Formalism for Specifying Compliant Motion Tasks”. In: *The International Journal of Robotics Research* 7.4 (1988), pp. 3–17.

[73] H. Bruyninckx and J. De Schutter. “Specification of force-controlled actions in the "task frame formalism"-a synthesis”. In: *IEEE Transactions on Robotics and Automation* 12.4 (Aug. 1996), pp. 581–589.

[74] Torsten Kröger et al. “Compliant motion programming: The task frame formalism revisited”. In: *Mechatronics & Robotics, Aachen, Germany* (2004).

[75] R. Smits et al. “iTASC: a tool for multi-sensor integration in robot manipulation”. In: *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. 2008, pp. 426–433.

[76] Joris De Schutter et al. “Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty”. In: *The International Journal of Robotics Research* 26.5 (2007), pp. 433–455.

[77] W. Decré, H. Bruyninckx, and J. De Schutter. “Extending the iTaSC Constraint-based Robot Task Specification Framework to Time-Independent Trajectories and User-Configurable Task Horizons”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 1941–1948.

[78] L. Halt et al. “Intuitive Constraint-Based Robot Programming for Robotic Assembly Tasks”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 520–526.

[79] F. Nägele et al. “Composition and Incremental Refinement of Skill Models for Robotic Assembly Tasks”. In: *2019 Third IEEE International Conference on Robotic Computing (IRC)*. 2019, pp. 177–182.

[80] E. Aertbeliën and J. De Schutter. “eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 1540–1546.

[81] Y. Pane et al. “A System Architecture for CAD-Based Robotic Assembly With Sensor-Based Skills”. In: *IEEE Transactions on Automation Science and Engineering* 17.3 (2020), pp. 1237–1249.

[82] C. A. Vergara P., J. De Schutter, and E. Aertbeliën. “Combining Imitation Learning With Constraint-Based Task Specification and Control”. In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 1892–1899.

[83] Michael E Tipping and Christopher M Bishop. “Probabilistic principal component analysis”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622.

[84] Bruce Donald et al. “Kinodynamic motion planning”. In: *Journal of the ACM (JACM)* 40.5 (1993), pp. 1048–1066.

[85] Steven M LaValle and James J Kuffner Jr. “Randomized kinodynamic planning”. In: *The international journal of robotics research* 20.5 (2001), pp. 378–400.

[86] David Hsu et al. “Randomized kinodynamic motion planning with moving obstacles”. In: *The International Journal of Robotics Research* 21.3 (2002), pp. 233–255.

[87] Calder Phillips-Grafflin and Dmitry Berenson. “Planning and resilient execution of policies for manipulation in contact with actuation uncertainty”. In: *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 752–767.

[88] F. Wirnshofer et al. “Robust, Compliant Assembly via Optimal Belief Space Planning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5436–5443.

[89] R. Kindel et al. “Kinodynamic motion planning amidst moving obstacles”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. 2000, pp. 537–543.

[90] F. Wirnshofer et al. “Robust, Compliant Assembly with Elastic Parts and Model Uncertainty”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 6044–6051.

[91] Zhenwang Yao and Kamal Gupta. "Path planning with general end-effector constraints". In: *Robotics and Autonomous Systems* 55.4 (2007), pp. 316–327.

[92] Adolfo Rodriguez et al. "Sampling-based path planning for geometrically-constrained objects". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 2074–2079.

[93] A. Rodrİguez, L. BasaÑez, and E. Celaya. "A Relational Positioning Methodology for Robot Task Specification and Execution". In: *IEEE Transactions on Robotics* 24.3 (2008), pp. 600–611.

[94] Rainer Jäkel. "Learning of Generalized Manipulation Strategies in Service Robotics". PhD thesis. 2013.

[95] Rüdiger Dillmann. "Teaching and learning of robot tasks via observation of human performance". In: *Robotics and Autonomous Systems* 47.2 (2004), pp. 109–116.

[96] Harish Ravichandar et al. "Recent Advances in Robot Learning from Demonstration". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1 (2020), null.

[97] Stefan Schaal. "Is imitation learning the route to humanoid robots?" In: *Trends in cognitive sciences* 3.6 (1999), pp. 233–242.

[98] Michael Bain and Claude Sammut. "A Framework for Behavioural Cloning." In: *Machine Intelligence 15*. 1995, pp. 103–129.

[99] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. "Active learning with statistical models". In: *Journal of artificial intelligence research* 4 (1996), pp. 129–145.

[100] Micha Hersch et al. "Dynamical system modulation for robot learning via kinesthetic demonstrations". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1463–1467.

[101] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[102] Zoubin Ghahramani and Michael I Jordan. "Supervised learning from incomplete data via an EM approach". In: *Advances in neural information processing systems*. 1994, pp. 120–127.

[103] Aude Billard et al. "Robot programming by demonstration". In: *Springer handbook of robotics*. Springer, 2008, pp. 1371–1394.

[104] S. Calinon et al. "Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework". In: *2009 9th IEEE-RAS International Conference on Humanoid Robots*. 2009, pp. 582–588.

[105] Leonel Rozo, Pablo Jiménez, and Carme Torras. "A robot learning from demonstration framework to perform force-based manipulation tasks". In: *Intelligent service robotics* 6.1 (2013), pp. 33–51.

[106] L. Rozo et al. "Learning Physical Collaborative Robot Behaviors From Human Demonstrations". In: *IEEE Transactions on Robotics* 32.3 (2016), pp. 513–527.

[107] Te Tang et al. "Autonomous alignment of peg and hole by force/torque measurement for robotic assembly". In: *2016 IEEE international conference on automation science and engineering (CASE)*. IEEE. 2016, pp. 162–167.

[108] Klas Kronander, Etienne Burdet, and Aude Billard. "Task transfer via collaborative manipulation for insertion assembly". In: *Workshop on human-robot interaction for industrial manufacturing, robotics, science and systems*. Citeseer. 2014.

[109] Lawrence R Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[110] Ali Al-Yacoub et al. "Symbolic-based recognition of contact states for learning assembly skills". In: *Frontiers in Robotics and AI* 6 (2019), p. 99.

[111] Shen Dong and Fazel Naghdy. "Application of hidden Markov model to acquisition of manipulation skills from haptic rendered virtual environment". In: *Robotics and Computer-Integrated Manufacturing* 23.3 (2007), pp. 351–360.

[112] A. J. Ijspeert, J. Nakanishi, and S. Schaal. "Movement imitation with nonlinear dynamical systems in humanoid robots". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. 2002, 1398–1403 vol.2.

[113] Auke Jan Ijspeert et al. "Dynamical movement primitives: learning attractor models for motor behaviors". In: *Neural computation* 25.2 (2013), pp. 328–373.

[114] Stefan Schaal. "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics". In: *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[115] H. Hoffmann et al. "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 2587–2592.

[116] P. Pastor et al. "Learning and generalization of motor skills by learning from demonstration". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 763–768.

[117] Guanwen Ding et al. "A Task-Learning Strategy for Robotic Assembly Tasks from Human Demonstrations". In: *Sensors* 20.19 (2020), p. 5505.

[118] P. Pastor et al. "Online movement adaptation based on previous sensor experiences". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 365–371.

[119] P. Pastor et al. "Towards Associative Skill Memories". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. 2012, pp. 309–315.

[120] Fares J Abu-Dakka et al. "Adaptation of manipulation skills in physical contact with the environment to reference force profiles". In: *Autonomous Robots* 39.2 (2015), pp. 199–217.

[121] B. Nemec et al. "Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile". In: *16th International Conference on Advanced Robotics (ICAR)*. Nov. 2013, pp. 1–7.

[122] Norbert Krüger et al. "Technologies for the Fast Set-Up of Automated Assembly Processes". In: *KI-Künstliche Intelligenz* 28.4 (2014), pp. 305–313.

[123] Aljaz Kramberger et al. "Generalization of orientation trajectories and force-torque profiles for robotic assembly". In: *Robotics and Autonomous Systems* 98 (2017), pp. 333–346.

[124] T. R. Savarimuthu et al. “Teaching a Robot the Semantics of Assembly Tasks”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* PP.99 (2017), pp. 1–23.

[125] A. Ude et al. “Orientation in Cartesian space dynamic movement primitives”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2997–3004.

[126] Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. “A survey of iterative learning control”. In: *IEEE control systems magazine* 26.3 (2006), pp. 96–114.

[127] Jens Kober, J Andrew Bagnell, and Jan Peters. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* (2013).

[128] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[129] David Silver et al. “Mastering the game of go without human knowledge”. In: *nature* 550.7676 (2017), pp. 354–359.

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

[131] Gergely Neu and Csaba Szepesvári. “Apprenticeship learning using inverse reinforcement learning and gradient methods”. In: *arXiv preprint arXiv:1206.5264* (2012).

[132] Vijaykumar Gullapalli, Roderic A Grupen, and Andrew G Barto. “Learning reactive admittance control”. In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. IEEE. 1992, pp. 1475–1480.

[133] V. Gullapalli, J. A. Franklin, and H. Benbrahim. “Acquiring robot skills via reinforcement learning”. In: *IEEE Control Systems* 14.1 (Feb. 1994), pp. 13–24.

[134] Levine, N. Wagener, and P. Abbeel. “Learning contact-rich manipulation skills with guided policy search”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 156–163.

[135] Sergey Levine and Pieter Abbeel. “Learning neural network policies with guided policy search under unknown dynamics”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 1071–1079.

[136] Weiwei Li and Emanuel Todorov. “Iterative linear quadratic regulator design for nonlinear biological movement systems.” In: *ICINCO (1)*. 2004, pp. 222–229.

[137] G. Thomas et al. “Learning Robotic Assembly from CAD”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1–9.

[138] T. Inoue et al. “Deep reinforcement learning for high precision assembly tasks”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 819–825.

[139] Gerrit Schoettler et al. “Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards”. In: *arXiv preprint arXiv:1906.05841* (2019).

[140] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *arXiv preprint arXiv:1801.01290* (2018).

[141] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[142] J. Ding, C. Wang, and C. Lu. "Transferable Trial-Minimizing Progressive Peg-in-hole Model". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2019, pp. 5862–5868.

[143] G. Hirzinger. "Direct Digital Robot Control using a Force-Torque Sensor". In: *IFAC Proceedings Volumes* 16.1 (1983). IFAC/IFIP Symposium on Real Time Digital Control Applications, Guadalajara, Mexico, 17-19 January 1983, pp. 243–255.

[144] S. Farsoni et al. "Compensation of Load Dynamics for Admittance Controlled Interactive Industrial Robots Using a Quaternion-Based Kalman Filter". In: *IEEE Robotics and Automation Letters* 2.2 (Apr. 2017), pp. 672–679.

[145] Nick Jakobi, Phil Husbands, and Inman Harvey. "Noise and the reality gap: The use of simulation in evolutionary robotics". In: *European Conference on Artificial Life*. Springer. 1995, pp. 704–720.

[146] T. N. Thulesen and H. G. Petersen. "RobWorkPhysicsEngine: A new dynamic simulation engine for manipulation actions". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2060–2067.

[147] Ellen Klingbeil, Samir Menon, and Oussama Khatib. "Experimental Analysis of Human Control Strategies in Contact Manipulation Tasks". In: *2016 International Symposium on Experimental Robotics*. Ed. by Dana Kulić et al. Cham: Springer International Publishing, 2017, pp. 275–286.

[148] Michael I Jordan and Robert A Jacobs. "Hierarchies of adaptive experts". In: *Advances in neural information processing systems*. 1992, pp. 985–992.

[149] R.C. Miall and D.M. Wolpert. "Forward Models for Physiological Motor Control". In: *Neural Networks* 9.8 (1996). Four Major Hypotheses in Neuroscience, pp. 1265–1279.

[150] Bin Fang et al. "Survey of imitation learning for robotic manipulation". In: *International Journal of Intelligent Robotics and Applications* (2019), pp. 1–8.

[151] Faraz Torabi, Garrett Warnell, and Peter Stone. "Behavioral cloning from observation". In: *arXiv preprint arXiv:1805.01954* (2018).

[152] Ajay Mandlekar et al. "Roboturk: A crowdsourcing platform for robotic skill learning through imitation". In: *arXiv preprint arXiv:1811.02790* (2018).

[153] Ralf Koeppe and Gerd Hirzinger. "Learning compliant motions by task-demonstration in virtual environments". In: *In Fourth International Symposium on Experimental Robotics*. Citeseer. 1995.

[154] H. Ogata and T. Takahashi. "Robotic assembly operation teaching in a virtual environment". In: *IEEE Transactions on Robotics and Automation* 10.3 (June 1994), pp. 391–399.

[155] John M Hsu and Steven C Peters. "Extending open dynamics engine for the DARPA virtual robotics challenge". In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer. 2014, pp. 37–48.

[156] OpenAI: Marcin Andrychowicz et al. "Learning dexterous in-hand manipulation". In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.

[157] Nathan P Koenig and Andrew Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2004, pp. 2149–2154.

[158] Kris Hauser. "Robust contact generation for robot simulation with unstructured meshes". In: *Robotics Research*. Springer, 2016, pp. 357–373.

[159] Olivier Michel. "Cyberbotics Ltd. Webots: professional mobile robot simulation". In: *International Journal of Advanced Robotic Systems* 1.1 (2004), p. 5.

[160] Carlo Pinciroli et al. "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems". In: *Swarm intelligence* 6.4 (2012), pp. 271–295.

[161] Eric Rohmer, Surya PN Singh, and Marc Freese. "V-REP: A versatile and scalable robot simulation framework". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1321–1326.

[162] Thomas Nicky Thulesen. "Dynamic simulation of manipulation & assembly actions". PhD thesis. University of Southern Denmark, 2015.

[163] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033.

[164] Russell Smith et al. "Open dynamics engine". In: (2005).

[165] Jyh-Ming Lien and Nancy M Amato. "Approximate convex decomposition of polyhedra". In: *Proceedings of the 2007 ACM symposium on Solid and physical modeling*. 2007, pp. 121–131.

[166] Anis Koubaasdfasdfj. *Robot operating system (ros): The complete reference*. Vol. 2. Springer, 2017.

[167] David Baraff. "Linear-time dynamics using Lagrange multipliers". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, pp. 137–146.

[168] Evan Drumwright et al. "Extending open dynamics engine for robotics simulation". In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer. 2010, pp. 38–50.

[169] Chrystopher L Nehaniv, Kerstin Dautenhahn, et al. "The correspondence problem". In: *Imitation in animals and artifacts* 41 (2002).

[170] Gabriel Robles-De-La-Torre and Vincent Hayward. "Force can overcome object geometry in the perception of shape through active touch". In: *Nature* 412.6845 (2001), pp. 445–448.

[171] Roland Sigrist et al. "Augmented visual, auditory, haptic, and multimodal feedback in motor learning: a review". In: *Psychonomic bulletin & review* 20.1 (2013), pp. 21–53.

[172] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[173] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[174] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "Understanding the exploding gradient problem". In: *ArXiv* abs/1211.5063 (2012).

[175] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12 (2000), pp. 2451–2471.

[176] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[177] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[178] I Sutskever, O Vinyals, and QV Le. "Sequence to sequence learning with neural networks". In: *Advances in NIPS* (2014).

[179] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[180] Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[181] A. Waibel et al. "Phoneme recognition using time-delay neural networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (1989), pp. 328–339.

[182] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". In: *arXiv preprint arXiv:1803.01271* (2018).

[183] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. "Regularizing and optimizing LSTM language models". In: *arXiv preprint arXiv:1708.02182* (2017).

[184] Victor Campos et al. "Skip rnn: Learning to skip state updates in recurrent neural networks". In: *arXiv preprint arXiv:1708.06834* (2017).

[185] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[186] Oriol Vinyals et al. "Show and tell: A neural image caption generator". In: *IEEE conference on computer vision and pattern recognition*. 2015, pp. 3156–3164.

[187] S. Scherzinger, A. Roennau, and R. Dillmann. "Contact Skill Imitation Learning for Robot-Independent Assembly Programming". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2019, pp. 4309–4316.

[188] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[189] Christopher M Bishop. *Mixture density networks*. Tech. rep. Citeseer, 1994.

[190] Stuart Russel et al. *Artificial intelligence: a modern approach*.

[191] Luka Crnkovic-Friis and Louise Crnkovic-Friis. "Generative choreography using deep learning". In: *arXiv preprint arXiv:1605.06921* (2016).

[192] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2017.

[193] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[194] Paul J Werbos. "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural networks* 1.4 (1988), pp. 339–356.

[195] Atilim Gunes Baydin et al. "Automatic Differentiation in Machine Learning: a Survey". In: *Journal of Machine Learning Research* 18.153 (2018), pp. 1–43.

[196] Nitish Shirish Keskar et al. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv preprint arXiv:1609.04836* (2016).

[197] Dominic Masters and Carlo Luschi. "Revisiting Small Batch Training for Deep Neural Networks". In: *CoRR* abs/1804.07612 (2018). arXiv: 1804.07612.

[198] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[199] George EP Box. "A note on the generation of random normal deviates". In: *Ann. Math. Stat.* 29 (1958), pp. 610–611.

[200] Joshua V Dillon et al. "Tensorflow distributions". In: *arXiv preprint arXiv:1711.10604* (2017).

[201] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2008.

[202] B Paul. "Analytical dynamics of mechanisms-a computer oriented overview". In: *Mechanism and Machine Theory* 10.6 (1975), pp. 481–507.

[203] R. Featherstone. "The Calculation of Robot Dynamics Using Articulated-Body Inertias". In: *The International Journal of Robotics Research* 2.1 (1983), pp. 13–30.

[204] C. A. Balafoutis, R. V. Patel, and P. Misra. "Efficient modeling and computation of manipulator dynamics using orthogonal Cartesian tensors". In: *IEEE Journal on Robotics and Automation* 4.6 (1988), pp. 665–676.

[205] S. McMillan and D. B. Orin. "Forward dynamics of multilegged vehicles using the composite rigid body method". In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. Vol. 1. 1998, 464–470 vol.1.

[206] Roy Featherstone. "Efficient Factorization of the Joint-Space Inertia Matrix for Branched Kinematic Trees". In: *The International Journal of Robotics Research* 24.6 (2005), pp. 487–500.

[207] J. M. Hollerbach. "A Recursive Lagrangian Formulation of Maniputator Dynamics and a Comparative Study of Dynamics Formulation Complexity". In: *IEEE Transactions on Systems, Man, and Cybernetics* 10.11 (1980), pp. 730–736.

[208] MH Raibert and BKP Horn. "Manipulator control using the configuration space method". In: *The Industrial Robot* 5.2 (1978), pp. 69–73.

[209] Chae H. An and John M. Hollerbach. "The Role of Dynamic Models in Cartesian Force Control of Manipulators". In: *The International Journal of Robotics Research* 8.4 (1989), pp. 51–72.

[210] Jun Nakanishi et al. "Operational Space Control: A Theoretical and Empirical Comparison". In: *The International Journal of Robotics Research* 27.6 (2008), pp. 737–757.

[211] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input". In: *Advanced Robotics* 25.5 (2011), pp. 581–603.

[212] Roy Featherstone and Oussama Khatib. “Load independence of the dynamically consistent inverse of the Jacobian matrix”. In: *The International Journal of Robotics Research* 16.2 (1997), pp. 168–170.

[213] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[214] Stefan Scherzinger, Arne Roennau, and Rüdiger Dillmann. *Virtual Forward Dynamics Models for Cartesian Robot Control*. 2020. arXiv: 2009.11888 [cs.RO].

[215] S. Scherzinger, A. Roennau, and R. Dillmann. “Inverse Kinematics with Forward Dynamics Solvers for Sampled Motion Tracking”. In: *19th International Conference on Advanced Robotics (ICAR)*. Dec. 2019, pp. 681–687.

[216] S. Scherzinger, A. Roennau, and R. Dillmann. “Forward Dynamics Compliance Control (FDCC): A new approach to cartesian compliance for robotic manipulators”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 4568–4575.

[217] Roy Featherstone. “An Empirical Study of the Joint Space Inertia Matrix”. In: *The International Journal of Robotics Research* 23.9 (2004), pp. 859–871.

[218] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4.

[219] Lester Miranda. “PySwarms: a research toolkit for particle swarm optimization in Python”. In: *Journal of Open Source Software* 3.21 (2018), p. 433.

[220] Georg Heppner et al. “FLA2IR—FLexible Automotive Assembly with Industrial Co-workers”. In: *Bringing Innovative Robotic Technologies from Research Labs to Industrial End-users: The Experience of the European Robotics Challenges*. Ed. by Fabrizio Caccavale et al. Cham: Springer International Publishing, 2020, pp. 97–126.

[221] M Kortman et al. “Building block based iBoss approach: fully modular systems with standard interface to enhance future satellites”. In: *66th International Astronautical Congress (Jerusalem)*. 2015, pp. 1–11.

[222] Dumitru Erhan et al. “Why does unsupervised pre-training help deep learning?” In: *Journal of Machine Learning Research* 11.Feb (2010), pp. 625–660.

[223] Alex Ellery, Joerg Kreisel, and Bernd Sommer. “The case for robotic on-orbit servicing of spacecraft: Spacecraft reliability is a myth”. In: *Acta Astronautica* 63.5 (2008), pp. 632–648.