Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Magne Johannes Erlandsen

Real-World Robot Learning Framework for Compliant Manipulation

Master's thesis in Cybernetics and Robotics (MTTK)
Supervisor: Jan Tommy Gravdahl
Co-supervisor: Akhil S. Anand
July 2023

NTNU
Norwegian University of
Science and Technology

Magne Johannes Erlandsen

Real-World Robot Learning Framework for Compliant Manipulation

Master's thesis in Cybernetics and Robotics (MTTK)
Supervisor: Jan Tommy Gravdahl
Co-supervisor: Akhil S. Anand
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This report concludes my master's thesis at the study Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The work conducted during this thesis is a continuation of my specialisation project (Erlandsen, 2023). For this reason, the following parts of this report are based on Erlandsen (2023):

- Sections 2.1 – 2.4 from chapter 2
- Sections 3.1 – 3.4 from chapter 3
- Tables B.1 and B.2 from Appendix B

Acknowledgements

I would like to take this opportunity to thank all the individuals that has been crucial in the completion of this Master's thesis.

First, I would like to thank my co-supervisor, Postdoctoral Fellow Akhil S. Anand, for always taking his time and effort to give guidance and thorough follow-up when requested, and for being supportive and solution-oriented throughout the semester.

I would also like to thank my supervisor, Professor Jan Tommy Gravdahl, for his availability and regular meetings every second week.

Furthermore, I want to thank my partner, for all her love and support. During both the highs and lows of this journey, her presence has constantly reminded me of what truly matters in life.

Finally, I want to thank my friends, for their great support and follow-up.

Abstract

This master's thesis explores the development of a real-world robot learning framework aimed at executing compliant robotic manipulation. Compliant manipulation is required in situations where the robot interacts with the environment and can improve safety, dexterity, flexibility, and energy efficiency. Variable impedance control (VIC) has emerged as a promising approach for this purpose, where the dynamic relationship between the robot and the environment is controlled at every time step.

In recent years, the stability analysis of VIC has advanced, enabling theoretical stability guarantees of such systems. Research has focused on learning how to vary the impedances, as this is a challenging but necessary task to fully utilise the flexibility that VIC offers. To address this problem, learning approaches like imitation learning, learning from demonstrations (LfD), or reinforcement learning (RL) have all been investigated by the research community. However, these approaches have primarily been utilised on simulated systems. Real-world robot systems have additional challenges related to communication, real-time constraints, hardware, and measurement noise; all contributing to conditions that are hard to resemble in simulators. To utilise learning approaches on real-world robots, a reliable and fast robot learning control framework is needed. Furthermore, the research conducted in this master's thesis aims to facilitate the use of a real-world robot variable impedance learning control (VILC) framework for the purpose of learning compliant robotic manipulation tasks.

Together with the specialisation project, this thesis demonstrates a real-world robot framework capable of learning and controlling the robot. A novel learning strategy involving kinaesthetic teaching and imitation learning is proposed to learn impedance profiles that can be fed to a variable impedance controller. The learning algorithm involves three steps: recording a desired end effector pose trajectory through kinaesthetic teaching, executing the recorded trajectory using impedance control, and setting up a supervised learning problem using a neural network to estimate the observed impedances from the execution data. By leveraging real-world robot data collected during experiments, the framework enhances learning for compliant robotic manipulation. The experiments are performed on the Franka Emika Panda robot, a versatile manipulator with seven degrees of freedom (DoF). These experiments using the robot learning framework demonstrate its value in assembly tasks, showcasing its potential for practical application in compliant manipulation scenarios. By contributing to the advancement of real-world robot learning frameworks, this research enriches the existing body of knowledge in the field of robotic control. The proposed methods and experimental findings provide valuable insights and potential advancements for future research in the area of robotic manipulation and learning.

Sammendrag

Denne masteroppgaven utforsker utviklingen av et læringsrammeverk for fysiske roboter med mål om å utføre fleksibel robotmanipulasjon. Fleksibel manipulasjon kreves i situasjoner hvor roboten interagerer med omgivelsene, og kan forbedre sikkerhet, fingerferdighet, fleksibilitet og energieffektivitet. Variabel impedanskontroll (VIC) har dukket opp som en lovende fremgangsmåte for dette formålet, hvor det dynamiske forholdet mellom roboten og omgivelsene styres hvert tidssteg.

I senere år, har stabilitetsanalysen av VIC utviklet seg, noe som muliggjør teoretiske stabilitetsgarantier av slike systemer. Videre har forskningen fokusert på å lære hvordan man varierer impedansene, da dette er en utfordrende, men nødvendig oppgave for å utnytte fleksibiliteten som VIC tilbyr fullt ut. For å takle dette problemet har læringsmetoder som imitasjonslæring, læring fra demonstrasjoner (LfD) eller forsterkende læring (RL) blitt undersøkt av forskningsmiljøet.

Imidlertid har disse metodene hovedsakelig blitt brukt på simulerte systemer. Virkelige robotsystemer har ekstra utfordringer knyttet til kommunikasjon, sanntidsbegrensninger, maskinvare og målefeil, som alle bidrar til forhold som er vanskelige å etterligne i simulatorer. For å anvende læringsmetodene på virkelige roboter, trengs et pålitelig og raskt læringsrammeverk for fysiske roboter. Forskningen som er gjennomført i denne masteroppgaven, har som mål å legge til rette for bruk av et variabel impedans læringskontroll (VILC) rammeverk for virkelige roboter med det overordnede formålet å lære samsvarende robotmanipulasjonsoppgaver.

Sammen med fordypningsprosjektet demonstrerer denne oppgaven et rammeverk for virkelige roboter som er i stand til læring og kontroll av roboten. En ny læringsstrategi som involverer kinestetisk læring og imitasjonslæring, er foreslått for å lære impedansprofiler som kan mates til en variabel impedanskontroller. Læringsalgoritmen innebærer tre steg: opptak av en ønsket endeeffektorbane ved hjelp av kinestetisk læring, utførelse av den registrerte banen ved bruk av impedanskontroll, og oppsett av et overvåket læringsproblem (eng. supervised learning problem) ved hjelp av et nevralt nettverk for å estimere de observerte impedansene fra utførelsesdataene. Ved å dra nytte av data fra virkelige roboter samlet under eksperimenter, forbedrer rammeverket læring for samhandlende robotstyring. Eksperimentene utføres på en Franka Emika Panda-robot, en allsidig manipulator med sju frihetsgrader. Disse eksperimentene, som er utført ved hjelp av robotlæringsrammeverket, viser dets verdi i monteringsoppgaver og dets potensiale for praktisk anvendelse i samhandlende manipulasjonsscenarioer. Ved å bidra til utviklingen av læringsrammeverk for virkelige roboter, beriker denne forskningen det eksisterende kunnskapsgrunnlaget innen feltet robotkontroll. De foreslåtte metodene og eksperimentelle funnene gir verdifull innsikt og potensielle fremskritt for fremtidig forskning innen robotmanipulasjon og læring.

Table of Contents

List of Tables

List of Algorithms

List of Figures

List of Acronyms

CPU central processing unit. xi, 41, 46

CUDA Compute Unified Device Architecture. 57

DH Denavit-Hartenberg. 7, 15

DoF degrees of freedom. v, 6, 8, 15

FCI Franka Control Interface. xiii, 3, 15, 17, 18

GPU graphical processing unit. xi, 19, 25, 41, 46, 57

LfD learning from demonstration. 22

ML machine learning. 13

MSE mean squared error. xi, 25, 39, 41, 43

PC personal computer. xvi, 17–19, 25, 57

protobuf Protocol Buffers. 18, 21, 45, 57

RL reinforcement learning. v, vii, 14, 49

ROS Robot Operating System. 3, 15, 17, 19, 57

SPD symmetric and positive definite. 8, 10, 12

VIC variable impedance control. v, vii, 2, 3, 5, 11–13, 22, 23, 31, 47, 49

VIL variable impedance learning. 3, 13, 31, 47, 49

VILC variable impedance learning control. v, vii, 13, 49

ZOH zero-order-hold. 21, 22

Glossary

Adam is a popular optimisation algorithm for training neural networks. 35

Control PC is the denotion of the realtime kernel computer connected by ethernet to the Panda robot. 18, 25, 57

Franka-interface is the low-level control interface that together with FrankaPy makes up a control framework for the Panda robot.. xi, xiii, xvi, 3, 15, 17–19, 21, 25, 45, 47, 57–61

FrankaPy is the high-level control interface, that together with Franka-interface makes up a control framework for the Panda robot.. xi, xiii, xvi, 3, 15, 18, 19, 21, 25, 45, 47, 57, 59–61

FrankaPy PC is the denotion of the computer running FrankaPy. 18, 19, 25, 57

libfranka is a software library that provides a programming interface for controlling the Panda robot. 18, 21

Panda robot is short for the Franka Emika Panda robot. xiii, xvi, 3, 15–18, 27, 47

Protocol Buffers is a free and open-source cross-platform data format used to serialise structured data.. xv

PyTorch is a machine learning framework used for applications such as computer vision and natural language processing. 19, 45, 46

zero-order-hold is to convert a discrete-time signal to a continuous-time signal by holding each sample value for one sample interval.. xv, 21, 22

1

Introduction

1.1 Problem Description and Motivation

Robotic manipulation tasks involving complex interactions with the environment pose a significant challenge in the field of robotics. This thesis aims to address the problem of enabling robotic manipulators to learn such manipulation skills efficiently. The primary objective of this thesis is to develop a robot learning framework tailored for torque-controlled robotic manipulators in real-world settings. The framework's key focus is on enabling the learning of manipulation skills directly from experiments conducted on real robotic manipulator systems, without having to rely on simulators.

The motivation behind using real-world robot data is to explore whether robots can resemble humans in learning manipulation skills without having access to large amounts of data. In simulations, experiments can be conducted faster than in real-time and in parallel to generate large amounts of data in a very short time. Whereas real-world robot data is expensive and challenging to obtain, therefore imposing limitations in terms of sample efficiency and time in learning. However, this approach offers the advantage of directly addressing potential modeling errors and discrepancies that arise due to the unique dynamics of each robot.

Most existing research in robot learning focuses on learning control strategies from simulations and subsequently transferring them to real robots. However, this approach often requires additional re-learning to account for the reality gap between simulations and real-world experiments. By exclusively utilising real-world robot data, this thesis aims to eliminate the need for a simulation framework, thereby simplifying the overall learning process.

It is important to emphasise that simulations have their own advantages and are widely used in many contexts. This thesis does not undermine the value of simulation-based learning; rather, it aims to investigate the potential of real-world robot data-driven approaches. Both simulation-based and real-world robot data-driven methods have their respective advantages and disadvantages, making it essential for the research community to explore and compare both approaches.

Learning compliance control is motivated by the need to enable robots and autonomous systems to interact safely and effectively with the dynamic and uncertain real world. Compliance control refers to the ability of a robot or machine to modulate its impedance, allowing it to adapt to varying environments and interact with objects and humans in a compliant and gentle manner. The traditional approach to robot control often relies on precise and rigid motions, which can be limiting when dealing with unstructured or changing environments. In contrast, compliance control enables robots to respond more gracefully to unexpected disturbances, uncertainties, and varying conditions. By acquiring compliance skills through learning, robots can effectively navigate complex and dynamic environments.

The following research questions will guide the research in this thesis:

1. How can a real-world robot learning framework be developed to facilitate the execution of compliant manipulation tasks?
2. How can existing machine learning techniques, such as deep reinforcement learning or imitation learning, be adapted and integrated into a real-world robot learning framework?
3. What are the potential advantages and disadvantages of utilising real-world robot data exclusively, as opposed to simulations, for learning compliant manipulation tasks, and how do these approaches impact the overall learning process?

By addressing these research questions within the proposed robot learning framework, this thesis seeks to contribute to the progress of robot learning research. The findings of this study will encourage further investigation into simulation-based and real-world robot data-driven approaches within the research community.

1.2 Contributions

The contributions of this thesis include

- Literature review exploring relevant topics within robotic control.
- Developing and testing a real-world robot framework that is suitable for learning complex robotic manipulation tasks.
- A novel impedance estimation method from real-world robot data variable impedance control (VIC).
- Experimental evaluation of the proposed framework and impedance estimation method on a robotic assembly task.

1.3 Scope and Delimitations

In this master's thesis, several delimitations have been established to define the boundaries and scope of the study. Firstly, the research is exclusively based on real-world robot data, and simulations are not considered. This ensures a focus on the practical applicability of

learning frameworks in real-world robotic manipulation tasks. The study centers on the Panda robot, assuming that the gained knowledge from this specific robotic arm extends to similar torque-controlled robots. Secondly, the primary focus for learning algorithms revolves around imitation learning and reinforcement learning. Other learning algorithms, including those based on haptic feedback, adaptive control, and Gaussian processes, are outside the scope of this thesis. Lastly, the analysis concentrates on various versions of impedance control, excluding consideration of other control approaches such as direct force control and position control. By clearly defining these delimitations, this thesis ensures a well-defined and focused exploration of real-world robot data-driven approaches and their applicability to specific robotic manipulation scenarios.

1.4 Structure of the Report

The structure of the report is presented to give an overview of the contents of this master's thesis. The report begins with a brief preface, acknowledgments, and abstract, the latter written in both English and Norwegian. Lists of tables, algorithms, and figures follows, providing a convenient reference for readers.

This chapter (Chapter 1) lays the foundation for the research by describing the problem and motivation behind developing a robot learning framework. The chapter outlines the contributions made in the thesis and defines the delimitations that set the scope of the research.

Chapter 2 delves into the theoretical background of the research. It covers preliminary concepts, robot dynamics, and the principles of impedance control. variable impedance control (VIC) and variable impedance learning (VIL) is discussed, along with challenges related to real-world robot learning.

Chapter 3 focuses on the specific hardware and software used in the experiments. The Franka Emika Panda robot is introduced, along with Robot Operating System (ROS) and Franka Control Interface (FCI). Details about the software environments, including Franka-interface, FrankaPy, and other related tools, are also provided.

Chapter 4 elaborates on the practical implementation aspects of impedance controllers. The development of the impedance profile and learning strategy is explained in detail, along with the setup of software environments.

Chapter 5 presents the results and findings from learning the impedance profile for robotic assembly tasks. Performance evaluation and metrics used to assess the learning framework are discussed.

Chapter 6 analyses and discusses the experimental results and the robot learning framework. Limitations of the research are addressed, and potential future work is proposed.

Chapter 7 provides a concise summary that concludes the key findings and contributions of this thesis.

Finally, appendices offer additional information, including details on code contributions and extra figures from experiments.

2

Background

In this chapter, the theoretical background needed for the rest of the thesis is presented. First, preliminaries and the robot dynamic equations are presented. Next, theory and literature reviews regarding impedance control and variable impedance control (VIC) are showcased. Finally, the chapter presents strategies and challenges for learning impedance profiles to effectively utilise VIC. The reader is reminded that Sections 2.1–2.4 is based on Erlandsen (2023), however 2.3 and 2.4 are expanded with thorough literature reviews.

2.1 Preliminaries

This section presents the preliminaries which are useful for derivations presented later in this chapter. First, the skew-symmetric operator is defined, which is useful for representing cross products. Next, rotation matrices and other useful representations of orientation in $\mathbb{R}^3$ will be presented.

Skew-symmetric operator

The skew symmetric operator $\boldsymbol{S} : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ may be defined as follows

$$\boldsymbol{S}\left(\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}\right) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{2.1}$$

which has the useful property $\boldsymbol{S}(\boldsymbol{u})\boldsymbol{v} = \boldsymbol{u} \times \boldsymbol{v}$.

Orientation and rotation

The orientation of coordinate frame *j* with respect to coordinate frame *i* can be expressed by denoting the basis vectors of frame *j* $(\boldsymbol{x}_j,\ \boldsymbol{y}_j,\ \boldsymbol{z}_j)$ in frame *i*. The result is what is

known as the rotation matrix $\boldsymbol{R}_j^i = (\boldsymbol{x}_j^i,\ \boldsymbol{y}_j^i,\ \boldsymbol{z}_j^i)$. The components of $\boldsymbol{R}_j^i$ is computed as

$$\boldsymbol{R}_j^i = \begin{bmatrix} \hat{\boldsymbol{x}}_j^i & \hat{\boldsymbol{y}}_j^i & \hat{\boldsymbol{z}}_j^i \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_j \cdot \boldsymbol{x}_i & \boldsymbol{y}_j \cdot \boldsymbol{x}_i & \boldsymbol{z}_j \cdot \boldsymbol{x}_i \\ \boldsymbol{x}_j \cdot \boldsymbol{y}_i & \boldsymbol{y}_j \cdot \boldsymbol{y}_i & \boldsymbol{z}_j \cdot \boldsymbol{y}_i \\ \boldsymbol{x}_j \cdot \boldsymbol{z}_i & \boldsymbol{y}_j \cdot \boldsymbol{z}_i & \boldsymbol{z}_j \cdot \boldsymbol{z}_i \end{bmatrix}. \tag{2.2}$$

It is worth noting that the rotation matrix $\boldsymbol{R}_j^i$ can be viewed both as an active rotation, moving a body attached to frame *i* to the orientation of frame *j* or as a passive rotation, changing basis of a vector expressed in frame *j* so that it instead is expressed in frame *i*. Both perspectives are equivalent, but these two interpretations can often be a source of confusion.

In addition to rotation matrices, orientations can be represented in several ways, including *Euler angles*, *Angle-axis* and *Quaternions*. In many cases it is helpful to use a more compact notation than rotation matrices, which consist of nine elements, as orientations in space only have three degrees of freedom (DoF). The Euler angle representation is a minimal representation including only three parameters, however, issues may arise due to its singularities – what is known as *gimbal lock* (Waldron and Schmiedeler, 2008). For this reason, The Euler angle representation will not be used excessively in this report.

The Angle-Axis representation defines a rotation via an angle θ and a unit vector $\boldsymbol{v}$ corresponding to the axis of rotation. Often such a representation is denoted compactly as the product $\theta \cdot \boldsymbol{v}$, which is called a *rotation vector*. Quaternions, that are based on the Angle-Axis representation, are defined as

$$\eta = \cos\left(\frac{\theta}{2}\right)\ ,\ \boldsymbol{\varepsilon} = \boldsymbol{v}\sin\left(\frac{\theta}{2}\right), \tag{2.3}$$

and is often denoted as a 4-vector:

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \boldsymbol{\varepsilon} \end{bmatrix}. \tag{2.4}$$

The quaternion product is defined as

$$\boldsymbol{r} = \boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = \begin{bmatrix} \eta_1\eta_2 - \boldsymbol{\varepsilon}_1^T\boldsymbol{\varepsilon}_2 \\ \eta_1\boldsymbol{\varepsilon}_2 + \eta_2\boldsymbol{\varepsilon}_1 + S(\boldsymbol{\varepsilon}_1)\boldsymbol{\varepsilon}_2 \end{bmatrix}, \tag{2.5}$$

where $S(\cdot)$ is the skew-symmetric operator from Equation 2.1. The inverse of a quaternion is given by

$$\boldsymbol{q}^{-1} = \begin{bmatrix} \eta \\ -\boldsymbol{\varepsilon} \end{bmatrix}. \tag{2.6}$$

The quaternion resulting from successive rotations can be computed via the quaternion product as $\boldsymbol{q}_{12} = \boldsymbol{q}_1 \otimes \boldsymbol{q}_2$, meaning the resulting quaternion after applying the rotations $\boldsymbol{q}_1$ followed by $\boldsymbol{q}_2$. Furthermore, the quaternion $\boldsymbol{q}^{-1} \otimes \boldsymbol{q} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$ corresponds to no rotation (Downs, 2003).

Given two quaternions $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$, representing two orientations relative some common base frame, the *difference quaternion* may be defined as

$$\boldsymbol{q}_{ji} = \boldsymbol{q}_i^{-1} \otimes \boldsymbol{q}_j = \begin{bmatrix} \eta_i\eta_j + \boldsymbol{\varepsilon}_i^T\boldsymbol{\varepsilon}_j \\ \eta_i\boldsymbol{\varepsilon}_j - \eta_j\boldsymbol{\varepsilon}_i + S(-\boldsymbol{\varepsilon}_i)\boldsymbol{\varepsilon}_j) \end{bmatrix}. \tag{2.7}$$

2.2 Robot Dynamics

The geometric jacobian of a robotic manipulator is defined as the relation between the joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ and the task-space end effector velocity in the following way

$$\boldsymbol{v}_e = \boldsymbol{J}\dot{\boldsymbol{q}}, \tag{2.8}$$

where $\boldsymbol{v}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \boldsymbol{\omega}_e \end{bmatrix}$, and $\dot{\boldsymbol{p}}_e$ is the translational velocity and $\boldsymbol{\omega}_e$ is the angular velocity. The jacobian $\boldsymbol{J}$ is a $6 \times n$ matrix, where n is the number of joints the manipulator has. The jacobian is in general configuration dependent, i.e., a function of the joints $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{q})$, and can be derived from the robot forward kinematics.

The forward kinematics is in principle to compute the position and orientation of the end effector given the joint angles (Waldron and Schmiedeler, 2008). The forward kinematics of the robot can be derived using the Denavit-Hartenberg (DH) convention.

The joint space robot dynamics are given by

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}_c - \boldsymbol{\tau}_e, \tag{2.9}$$

where $\boldsymbol{H}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is the vector of Coriolis and centrifugal torques, $\boldsymbol{g}(\boldsymbol{q})$ is the vector of gravitational torques, $\boldsymbol{\tau}_c$ is the vector of input control torques, and $\boldsymbol{\tau}_e$ is the vector of torques applied by the manipulator to the environment.

The Cartesian end effector velocity $\boldsymbol{v}_e$, is computed through the Jacobian as described by Equation 2.8. In the case of $n = 6$, i.e., a non-redundant and non-singular manipulator, the task space robot dynamics are given by

$$\boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{v}}_e + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{v}_e + \boldsymbol{\eta}(\boldsymbol{q}) = \boldsymbol{h}_c - \boldsymbol{h}_e, \tag{2.10}$$

where $\boldsymbol{\Lambda}(\boldsymbol{q}) = (\boldsymbol{J}\boldsymbol{H}(\boldsymbol{q})^{-1}\boldsymbol{J}^T)^{-1} \in \mathbb{R}^{6\times6}$ is the task space inertia matrix, $\boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{J}^{-T}\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{J}^{-1} - \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{J}}\boldsymbol{J}^{-1} \in \mathbb{R}^{6\times6}$ is the task space Coriolis and centrifugal matrix, $\boldsymbol{\eta}(\boldsymbol{q}) = \boldsymbol{J}^{-T}\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^6$ gravitational force, $\boldsymbol{h}_c = \boldsymbol{J}^{-T}\boldsymbol{\tau}_c$ is the end effector wrench corresponding to the input control torques $\boldsymbol{\tau}_c$, and $\boldsymbol{h}_e = \begin{bmatrix} \boldsymbol{f}_e^T & \boldsymbol{m}_e^T \end{bmatrix}^T$ is the end effector wrench, i.e. the force and moment applied by the end effector to the environment. The operator $(\cdot)^{-T}$ means inverse transposed, i.e. $((\cdot)^{-1})^T$. (Villani and Schutter, 2008, p. 164)

2.3 Impedance Control

Controlling robotic manipulators in physical contact with the environment has posed a significant challenge for researchers over many decades. This ongoing pursuit has resulted in a wide range of control algorithms and specialised hardware designs. In contrast, controlling a manipulator without contact, referred to as motion control, is now mostly considered resolved, with current research focused on optimising established frameworks such as inverse dynamics control, adaptive motion control, and independent joint PID control (Slotine and Li, 1987).

However, the research on controlling robotic manipulators during physical contacts has recently centered around a specific class of control methods, namely impedance control

(Hogan, 1985a). This class has emerged as a suitable control architecture for improving the performance of robotic manipulation in situations where traditional control approaches tend to fail. Examples include tasks involving uncertain pose estimation of manipulated objects, physical contact, and tasks requiring appropriate responses to unexpected disturbances (Villani and Schutter, 2008).

Historically, there has been several attempts of developing control strategies aimed at controlling robots in contact. An alternative approach called stiffness control (Salisbury, 1980) allowed task designers to specify a desired trajectory and corresponding stiffness without explicit force control. By intentionally offsetting the reference trajectory inside the object to be contacted, desired contact forces could be achieved. This technique did not require separating the task space into orthogonal position- and force-controlled subspaces, and could be implemented without a force sensor on torque-controlled robots (Villani and Schutter, 2008, p. 167). Hogan expanded upon this concept and developed an impedance model describing the relationship between physical effort (force) and flow (velocity) as a second-order linear dynamical system parameterised by virtual inertia, damping, and stiffness (Hogan, 1985a).

It is crucial to view impedance control not as a specific control implementation but as a concept that can be implemented in various ways. Hogan's proposed implementation scheme (Hogan, 1985b) is applicable to torque-controlled manipulators. An alternative approach known as admittance control reverses the causality of impedance by utilising force measurements to simulate a trajectory based on the desired impedance model. This trajectory can then be fed into a high-performance position controller. The implementation with impedance causality offers better robustness in rigid contact, while the admittance control scheme is more widely applicable and provides better nominal performance in free motion (Ott et al., 2010). (Kronander, 2015, p. 11–13)

Given the task space robot dynamics in Equation 2.10, impedance control is obtained via the control law

$$\boldsymbol{h}_c = \boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{\alpha} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{v}_e + \boldsymbol{\eta}(\boldsymbol{q}) + \boldsymbol{h}_e, \tag{2.11}$$

where $\boldsymbol{\alpha}$ is to be determined. This control law results in the following dynamics

$$\dot{\boldsymbol{v}}_e = \boldsymbol{\alpha},$$

where the end effector acceleration $\dot{\boldsymbol{v}}_e$ is controlled directly. Given a desired acceleration $\dot{\boldsymbol{v}}_d$ and a desired velocity $\boldsymbol{v}_d$, $\boldsymbol{\alpha}$ is chosen as

$$\boldsymbol{\alpha} = \dot{\boldsymbol{v}}_d + \boldsymbol{K}_M^{-1}(\boldsymbol{K}_D\Delta\boldsymbol{v}_{de} + \boldsymbol{K}_P\Delta\boldsymbol{x}_{de} - \boldsymbol{h}_e), \tag{2.12}$$

where $\boldsymbol{K}_M$, $\boldsymbol{K}_D$ and $\boldsymbol{K}_P$ are 6×6 symmetric and positive definite (SPD) matrices, $\Delta\boldsymbol{p}_{de} = \boldsymbol{p}_d - \boldsymbol{p}_e$ is the generalised position error, and $\Delta\boldsymbol{v}_{de} = \boldsymbol{v}_d - \boldsymbol{v}_e$ is the generalised velocity error. The resulting closed-loop dynamics become

$$\boldsymbol{K}_M\Delta\dot{\boldsymbol{v}}_{de} + \boldsymbol{K}_D\Delta\boldsymbol{v}_{de} + \boldsymbol{K}_P\Delta\boldsymbol{x}_{de} = \boldsymbol{h}_e, \tag{2.13}$$

i.e., the dynamic relationship between the robot and the environment is modelled as a 6-DoF mass-spring-damper system. (Anand et al., 2022)

A block diagram of impedance control with $\dot{\boldsymbol{v}}_d = \boldsymbol{v}_d = \boldsymbol{0}$ is shown in Figure 2.1, however, with slightly different notation. This diagram is adapted from Abu-Dakka and Saveriano (2020).

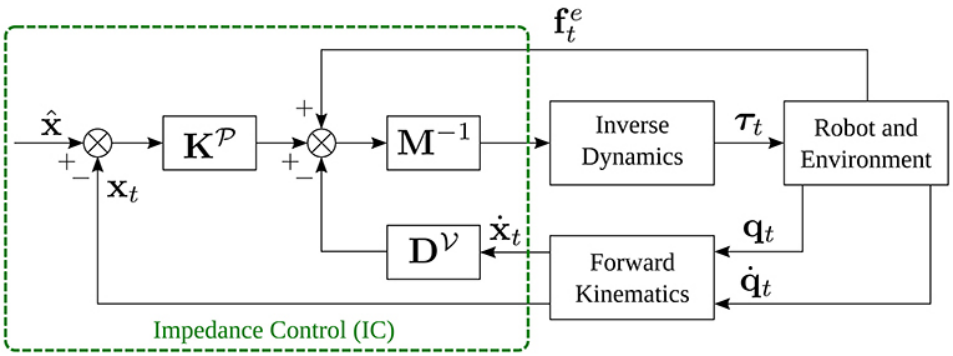


Figure 2.1: Block diagram of impedance control with desired acceleration and velocity set to zero. The figure is adapted from Abu-Dakka and Saveriano (2020).

Simplified impedance control scheme

The controller consisting of Equation 2.11 and Equation 2.12 includes what is called *inertia shaping*, as this controller tries to control the apparent inertia $\boldsymbol{K}_M$ of the interaction between the robot and the environment. This inertia shaping is indicated in Equation 2.13 as in addition to the damping and stiffness matrices, $\boldsymbol{K}_D$ and $\boldsymbol{K}_P$, also the apparent inertia matrix $\boldsymbol{K}_M$ can be chosen freely. A simplified version of this impedance controller, without inertia shaping, can be obtained by setting the apparent inertia in Equation 2.12 equal to the actual robot inertia, i.e., $\boldsymbol{K}_M := \boldsymbol{\Lambda}(\boldsymbol{q})$. If the resulting equation for $\boldsymbol{\alpha}$ is substituted into Equation 2.11 the control law becomes

$$\boldsymbol{h}_c = \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{v}}_d + \boldsymbol{K}_D\Delta\boldsymbol{v}_{de} + \boldsymbol{K}_P\Delta\boldsymbol{x}_{de} + \boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{v}_e + \boldsymbol{\eta}(\boldsymbol{q}), \tag{2.14}$$

where the external wrench $\boldsymbol{h}_e$ is no longer needed in the control law, as opposed to the inertia shaping controller. This simplified control scheme corresponds to stiffness control (Villani and Schutter, 2008) with the inclusion of coriolis compensation.

The closed-loop dynamics of the impedance controller *without* inertia shaping become

$$\boldsymbol{\Lambda}(\boldsymbol{q})\Delta\dot{\boldsymbol{v}}_{de} + \boldsymbol{K}_D\Delta\boldsymbol{v}_{de} + \boldsymbol{K}_P\Delta\boldsymbol{x}_{de} = \boldsymbol{h}_e, \tag{2.15}$$

i.e., Equation 2.13 with $\boldsymbol{K}_M = \boldsymbol{\Lambda}(\boldsymbol{q})$.

Alternative approach for impedance control

Villani and Schutter (2008) describes an alternative implementation of impedance control, where the end effector wrench error $\boldsymbol{h}_\Delta$ is determined more carefully to ensure a geometrically consistent active stiffness (Villani and Schutter, 2008). This approach is included for completeness, but is not utilised in this thesis. Furthermore, this control scheme defines impedances in the end effector frame, while in the previous method impedances were defined in the robot's inertial base frame. This alternative implementation is obtained from Equation 2.11, but with $\boldsymbol{\alpha}$ chosen as

$$\boldsymbol{\alpha} = \dot{\boldsymbol{v}}_d + \boldsymbol{K}_M^{-1}(\boldsymbol{K}_D\Delta\boldsymbol{v}_{de} + \boldsymbol{h}_\Delta - \boldsymbol{h}_e), \tag{2.16}$$

where the end effector wrench error $\boldsymbol{h}_\Delta$ is given by the equations

$$\begin{aligned} \boldsymbol{h}_\Delta &= \boldsymbol{h}_{\Delta t} + \boldsymbol{h}_{\Delta o}, \\ \text{with } \boldsymbol{h}_{\Delta t} &= \begin{bmatrix} \boldsymbol{f}_{\Delta t} \\ \boldsymbol{m}_{\Delta t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}'_{Pt} \Delta \boldsymbol{p}_{de} \\ \boldsymbol{K}''_{Pt} \Delta \boldsymbol{p}_{de} \end{bmatrix}, \\ \text{and } \boldsymbol{h}_{\Delta o} &= \begin{bmatrix} \boldsymbol{0}_{3\times 1} \\ \boldsymbol{m}_{\Delta o} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_{3\times 1} \\ \boldsymbol{K}'_{Po} \boldsymbol{\varepsilon}_{de} \end{bmatrix}, \end{aligned} \tag{2.17}$$

where $\Delta \boldsymbol{p}_{de} = \boldsymbol{p}_d - \boldsymbol{p}_e \in \mathbb{R}^{3\times 3}$ is the offset between the desired position and the current end effector position, and $\boldsymbol{\varepsilon}_{de} \in \mathbb{R}^{3\times 3}$ is the vector part of the difference quaternion between $\boldsymbol{q}_d$ and $\boldsymbol{q}_e$ as described in Equation 2.7.

The gain-matrices $\boldsymbol{K}'_{Pt},\ \boldsymbol{K}''_{Pt},\ \boldsymbol{K}'_{Po} \in \mathbb{R}^{3\times 3}$ are given by

$$\begin{aligned} \boldsymbol{K}'_{Pt} &= \frac{1}{2} \boldsymbol{R}_d \boldsymbol{K}_{Pt} \boldsymbol{R}_d^T + \frac{1}{2} \boldsymbol{R}_e \boldsymbol{K}_{Pt} \boldsymbol{R}_e^T, \\ \boldsymbol{K}''_{Pt} &= \frac{1}{2} \boldsymbol{S}(\Delta \boldsymbol{p}_{de}) \boldsymbol{R}_d \boldsymbol{K}_{Pt} \boldsymbol{R}_d^T, \\ \boldsymbol{K}'_{Po} &= 2 \boldsymbol{E}^T(\eta_{de}, \boldsymbol{\varepsilon}_{de}) \boldsymbol{R}_e \boldsymbol{K}_{Po} \boldsymbol{R}_e^T, \end{aligned} \tag{2.18}$$

with $\boldsymbol{E}(\eta_{de}, \boldsymbol{\varepsilon}_{de}) = \eta_{de} I - \boldsymbol{S}(\boldsymbol{\varepsilon}_{de})$ where $\boldsymbol{S}(\cdot)$ is the skew-symmetric operator as defined in Equation 2.1. $\boldsymbol{K}_{Pt}$ and $\boldsymbol{K}_{Po}$ are 3×3 SPD matrices corresponding to translational and rotational stiffness respectively (Villani and Schutter, 2008).

2.4 Variable Impedance Control

Impedance control with constant impedance parameters can be a suitable solution in certain cases. However, the ability to vary the impedance parameters during a task provides greater flexibility and can have a significant impact on performance across a wide range of tasks. Inserting a key into a keyhole is an example of a human task where varying the impedance, in this case the muscular stiffness, is essential to complete the task successfully. Initially, a person may hold the key with a relatively relaxed grip, allowing for smooth insertion without excessive force. However, as the key enters the keyhole, they need to adjust the impedance in their fingers and hand to provide enough resistance and stability for accurate placement. As the key approaches the locking mechanism, the person may need to exert more force and increase the muscular stiffness in their fingers, hand, and wrist to overcome any resistance or misalignment. This adjustment allows them to turn the key smoothly and engage the lock mechanism.

In the field of biological motor control, it is well-known that humans employ varying impedance in a task-dependent manner (Burdet et al., 2001; Gomi and Osu, 1998). Extensive research efforts have focused on assigning varying impedance to achieve various objectives, such as improved performance, enhanced safety, or reduced energy consumption. However, the stability implications of allowing the impedance parameters to vary have until recently been largely overlooked by the research community. Notably, Yang et al. (2011) proposed a biologically inspired varying impedance controller and rigorously

proved its stability. However, for the general case of time-varying impedance, the standard energy-based stability analysis is no longer valid when considering arbitrarily varying stiffness matrices (Kronander, 2015, p. 81).

Despite overlooking stability challenges, variable impedance control has demonstrated success in numerous applications, indicating that reasonable impedance variations are generally sufficiently small or slow for a wide range of tasks. However, these notions provide limited guidance when designing varying impedance profiles since the required speed or magnitude of variations cannot be determined before task execution.

Hannaford and Ryu (2002) and Ryu et al. (2004) utilised time-domain passivity, which is based on online computation of energy quantities and injection of damping. The passivity layer (PL) approach introduced in Franken et al. (2011) also relies on time-domain passivity to address interactivity and delay in bilateral telemanipulation. This approach leverages the ability to precisely compute energy exchange between a robot and its environment, provided the robot exhibits impedance causality (velocity in, force out)[1], as demonstrated in Stramigioli et al. (2005).

Ferraguti et al. (2013) proposed a similar approach specifically for the case of varying stiffness matrices in VIC. Their controller monitors the system's energy online, but stabilisation is achieved by reverting to a constant stiffness component rather than introducing additional damping.

While the aforementioned methods can theoretically stabilise a system under variable impedance control with an arbitrary varying stiffness profile, they either introduce damping (Ryu et al., 2004) or modify the stiffness profile (Ferraguti et al., 2013) based on the observed state trajectory during task execution. Consequently, it is impossible to determine the exact impedance profile that a robot will utilise prior to task execution. As a result, even if a task-specific impedance profile is carefully designed, the robot may show a completely different behaviour than intended during task execution. Kronander (2015) introduced more general stability conditions for variable impedance control. As opposed to previous stability conditions, these conditions are *state-independent* and may thus be evaluated before task execution. (Kronander, 2015, p. 14–15)

The concept of imposing constraints on variable impedance matrices to ensure stability prior to execution has been further developed by Sun et al. (2019), who introduced novel constraints on variable impedance matrices. These constraints guarantees exponential stability while simultaneously ensuring boundedness of the robot's position, velocity, and acceleration. Later, Spyrakos-Papastavridis et al. (2020) proposed a Passivity-Preservation Control (PPC) approach, which facilitates the implementation of stable VIC. Furthermore, they presented both joint and Cartesian space variants of the PPC controller, allowing for intuitive definition of interaction tasks. Finally, Park and Choi (2020) proposed sufficient conditions for ensuring input-to-state stability (ISS) for VIC.

Going back to the mathematical description of impedance control from (2.11), (2.12) and (2.13), VIC is obtained by varying the impedance matrices $\boldsymbol{K}_P$, $\boldsymbol{K}_D$, and, in the case of inertia shaping, $\boldsymbol{K}_M$. For completeness Kronander's sufficient stability conditions (Kronander, 2015, p. 82–83) is summarised by the following theorem, which is rewritten to match the notation in this thesis:

[1] This impedance causality matches the physical impedance described in Hogan (1985a).

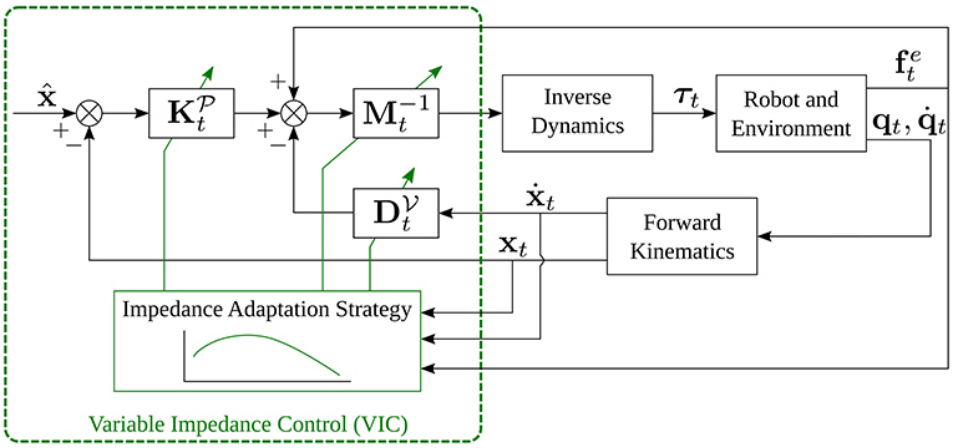


Figure 2.2: Block diagram of variable impedance control (VIC) with desired acceleration and velocity set to zero. The figure is adapted from Abu-Dakka and Saveriano (2020).

Theorem 1 (Stability conditions under dynamic decoupling)**.** *Let $\boldsymbol{K}_M$ be a constant, symmetric and positive definite (SPD) matrix. Let $\boldsymbol{K}_P(t)$ and $\boldsymbol{K}_D(t)$ be SPD and continuously differentiable varying stiffness and damping profiles. Then, the system in eq.* (2.13) *with $\boldsymbol{h}_e = 0$ is globally uniformly stable if there exists an $\alpha > 0$ such that $\forall t \geq 0$:*

1. $\alpha \boldsymbol{K}_M - \boldsymbol{K}_D(t) \leq 0$ *(negative semi-definite)*

2. $\dot{\boldsymbol{K}}_P(t) + \alpha \dot{\boldsymbol{K}}_D(t) - 2\alpha \boldsymbol{K}_P(t) \leq 0$ *(negative semi-definite)*

If in 2) semi-definiteness is replaced with definiteness, the stability property is in addition asymptotic.

The stability analysis of Theorem 1 can be drastically simplified under certain assumptions. For example, in the case of diagonal inertia, damping and stiffness matrices, and constant damping ratio, i.e. $d_x(t) := 2\zeta\sqrt{m_x k_x(t)}$, the stability conditions would reduce to (in x-direction):

$$\dot{k}_x(t) < \frac{2\alpha\sqrt{k_x(t)}^3}{\sqrt{k_x(t)} + \zeta\alpha\sqrt{m_x}},$$

where ζ is the constant damping ratio, and m_x and $k_x(t)$ are the inertia and stiffness values in x-direction. The conditions would be likewise in y- and z-direction, included in the rotational components.

A block diagram of variable impedance control (VIC) with $\dot{\boldsymbol{v}}_d = \boldsymbol{v}_d = \mathbf{0}$ is shown in Figure 2.2, again with slightly different notation. This diagram is adapted from Abu-Dakka and Saveriano (2020).

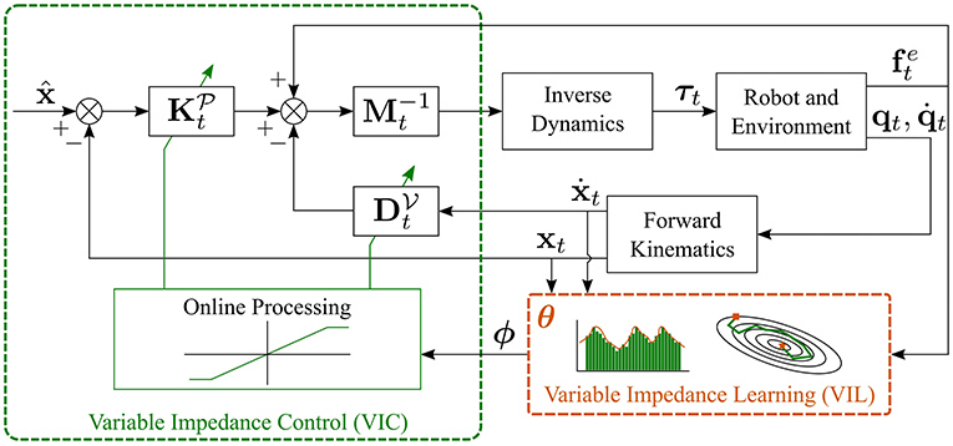


Figure 2.3: Block diagram of variable impedance learning (VIL) with desired acceleration and velocity set to zero. The figure is adapted from Abu-Dakka and Saveriano (2020).

2.5 Variable Impedance Learning

As specifying the impedances in VIC is difficult to do manually, it can be a suitable task for a machine learning (ML) algorithm. Learning such impedance profiles is referred to as variable impedance learning (VIL).

Abu-Dakka and Saveriano (2020) differs between VIL and variable impedance learning control (VILC). VIL focuses on learning algorithms used to encode variable impedance gains for tasks requiring adaptable stiffness and damping in robots. The learning process involves training data provided by an expert user, typically through kinaesthetic teaching, to acquire parameterised impedance gains represented by a set of parameters. Using this data, the learning algorithm approximates nonlinear mappings to obtain desired variable stiffness and damping matrices based on measurements of position, velocity, and force. During runtime, the learned model is used to retrieve the desired variable impedance gains, which are then employed to achieve desired closed-loop properties in robot interactions.

A block diagram of VIL with $\dot{\boldsymbol{v}}_d = \boldsymbol{v}_d = \mathbf{0}$ is shown in Figure 2.3. Again, the diagram is adapted from Abu-Dakka and Saveriano (2020), and uses slightly different notation compared to this thesis.

On the other hand, VILC extends beyond the scope of VIL by merging the learning algorithm with the underlying control structure. This integration allows for a more intricate impedance learning process, often involving iterative updates and robot self-exploration. Unlike VIL, VILC's data collection for learning impedance gains depends on the specific control strategy being used, and both learning and control blocks are combined. VILC also goes a step further by updating the target pose or reference trajectory alongside the impedance gains, offering more complex impedance learning behaviours.

A block diagram of the variable impedance learning control with $\dot{\boldsymbol{v}}_d = \boldsymbol{v}_d = \mathbf{0}$ is shown in Figure 2.4. Again, the diagram is adapted from Abu-Dakka and Saveriano (2020), and uses slightly different notation compared to this thesis.

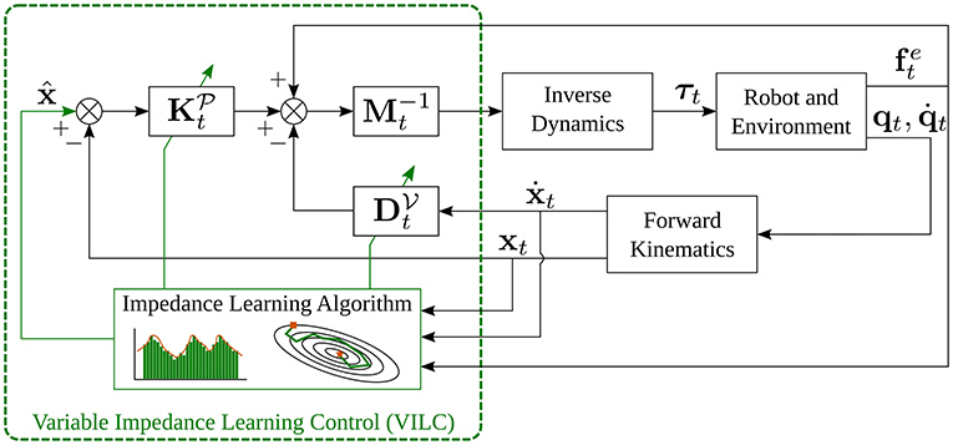


Figure 2.4: Block diagram of variable impedance learning control (VILC) with desired acceleration and velocity set to zero. The figure is adapted from Abu-Dakka and Saveriano (2020).

2.6 Challenges with Reinforcement Learning on Real-World Robots

Reinforcement learning (RL) holds promise for addressing complex robotic tasks, but its application with real-world robots has been hindered by challenges and unreliability. In Mahmood et al. (2018a), a learning task is developed with a UR5 robotic arm to highlight the importance of task setup and its impact on learning performance. The research emphasises that oversights in setup details can make learning, reproducibility, and fair comparisons difficult, but offers mitigating steps to overcome these challenges. The study demonstrates the potential for extensive RL research based on real-world robots by achieving highly reliable and repeatable experiments. Next, Mahmood et al. (2018b) addresses the challenges of applying RL to physical robots and emphasises the need for benchmark tasks and supporting source code. The paper highlights the importance of carefully setting up the task interface and computations for successful RL implementation on physical robots. It also discusses the sensitivity of modern learning algorithms to hyperparameters and the need for task-specific tuning to achieve optimal performance. Finally, Dulac-Arnold et al. (2021) identifies and formalises a series of independent challenges related to RL, which are then analysed on state-of-the-art learning algorithms. The paper also introduces a suite of continuous control environments called ”realworldrl-suite”, which serves as an open-source benchmark for evaluating RL algorithms in real-world scenarios.

3

Hardware and software

In this chapter the relevant robot hardware and software will be presented. The hardware used in this project is the Franka Emika Panda robot (hereafter referred to as *the Panda robot*) – a robotic arm with seven degrees of freedom (DoF). The software consists of two main modules; FrankaPy and Franka-interface; which are built upon Robot Operating System (ROS) and Franka Control Interface (FCI). The reader is reminded that Sections 3.1–3.4 are based on Erlandsen (2023).

3.1 Franka Emika Panda Robot

The Franka Emika Panda robot is, as mentioned, a 7-DoF robotic arm. It has seven revolute joints, each attached with a torque sensor, which makes it able to sense the impact of any external forces. The Panda robot is relatively cheap compared to other robots in its class. In 2017 the team behind this robot was awarded the *German Future Prize*, because the robot was "*cheap, affordable and easy to use*" (Deutche Welle, 2017).

The Panda robot's kinematic chain is indicated by Figure 3.1, and the corresponding DH parameters are shown in Table 3.1 (Franka Emika, b). Without going into details, it can be mentioned that these DH parameters follow Craig's convention and not the classical DH convention. This is equivalent to the classical convention, however, each frame is assigned differently in Figure 3.1 compared to what would be the case when using the classical DH convention.

The Panda robot is equipped with joint torque sensors but does not have an end effector force-torque sensor.

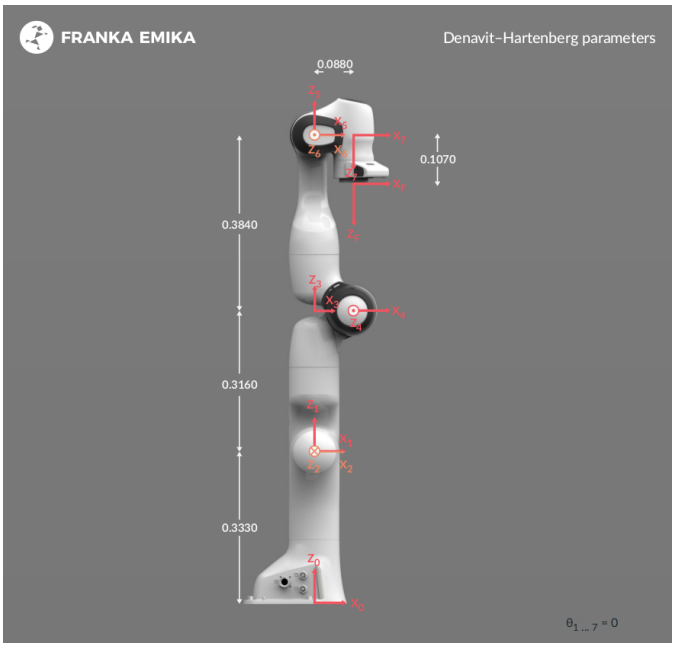


Figure 3.1: The Panda robot's kinematic chain

Joint	$\boldsymbol{\theta}$ (rad)	$\boldsymbol{d}$ (m)	$\boldsymbol{a}$ (rad)	$\boldsymbol{\alpha}$ (rad)
Joint 1	θ_1	0.333	0	0
Joint 2	θ_2	0	0	$-\pi/2$
Joint 3	θ_3	0.316	0	$\pi/2$
Joint 4	θ_4	0	0.0825	$\pi/2$
Joint 5	θ_5	0.384	-0.0825	$-\pi/2$
Joint 6	θ_6	0	0	$\pi/2$
Join 7	θ_7	0	0.088	$\pi/2$
Flange	0	0.107	0	0
Franka Hand	0	0.1034	0	0

Table 3.1: The DH parameters for the Panda robot

3.2 Robot Operating System

Robot Operating System (ROS) is an open source framework which provides tools to help software developers create robot applications. The framework includes hardware abstraction, device drivers, libraries, visualisers, message-passing, package management, and more. In this project ROS Noetic is used, which is mainly made for Ubuntu 20.04 (Stanford Artificial Intelligence Laboratory et al.).

ROS is utilised in this thesis for various purposes. We use the rosbag tool for recording data, message-passing for exchanging parameters, and the ROS clock for managing the control frequency by incorporating appropriate sleep commands. Additionally, we rely on the rospy package, which is a Python framework for ROS.

Saving Robot States with Rosbag

In the context of controlling the Panda robot an important aspect is the need to save robot states efficiently. Initial attempts to save the data within the robot framework proved inadequate due to the real-time performance requirements. Consequently, a more efficient approach was sought. *Rosbag* emerged as a promising solution as it allows saving all required data in a deserialised fashion, leveraging ROS messages. This capability enables the recording of selected ROS topics. Once the recording is completed, the data is saved to a ".bag"-file. This file can be opened and transformed into more readable data structures afterwards, such as converting serialised binary data into a Python dictionary.

3.3 Franka Control Interface

The Franka Control Interface (FCI) allows a fast and direct low-level bidirectional connection to the Panda robot. It provides the current status of the robot and enables its direct control with an external workstation PC connected via ethernet. By using the open source C++ library *libfranka*, real-time communication at 1 kHz is possible. This communication includes sending control commands, receiving measurements of joint position, velocities and torques, receiving estimation of external forces and torques, and receiving the forward kinematics and Jacobian matrix for all joints. In addition, *franka_ros* connects the Panda robot with the entire ROS ecosystem and integrates libfranka into ROS Control (Franka Emika, 2017). However, *franka_ros* is not used directly in this project, as Franka-interface is used for ROS integration instead. A schematic overview of the communication with FCI is shown in Figure 3.2.

The Franka Hand, a hand attachment for the Panda robot, operates in a non-real-time manner, while the robot arm operates in real-time.

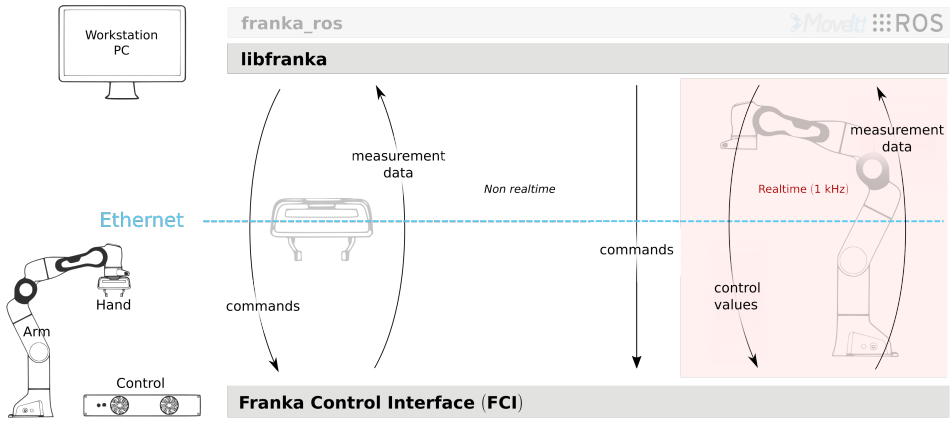


Figure 3.2: Schematic overview of the communication with the robot through FCI and *libfranka* ((Franka Emika, a))

3.4 Franka-interface and FrankaPy

Franka-interface and FrankaPy make up a a modular robotic control stack that provides a customisable and accessible interface to the Panda robot (Zhang et al., 2020). This framework abstracts high-level robot control commands as skills, which are decomposed into combinations of trajectory generators, feedback controllers, and termination handlers. Low-level control is implemented in C++ and can run at 1kHz, and high-level commands are exposed in Python. In addition, external sensor feedback, like estimated object poses, can be streamed to the low-level controllers in real time.

Figure 3.3 shows how the different modules communicate. Franka-interface uses libfranka so that real-time communication at 1kHz is still possible (Zhang et al., 2020).

FrankaPy relies on Google protobuf for communication between the different modules. It utilises the client-server model, where the server runs on the Control PC, and the client, in this case, the FrankaPy PC, communicates with the server. The server publishes

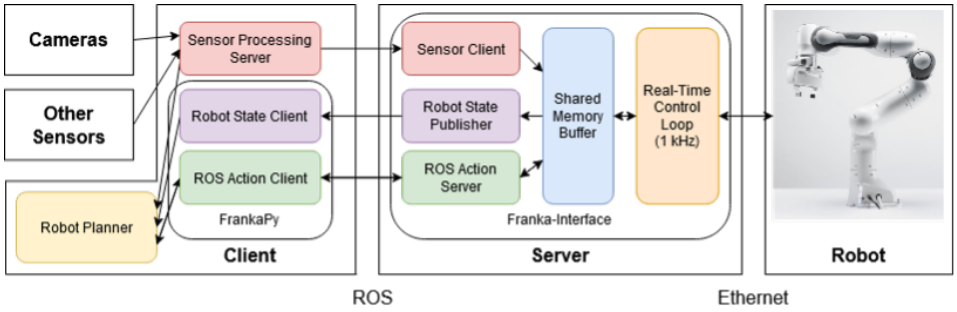


Figure 3.3: System Diagram for FrankaPy and Franka-interface

states to FrankaPy, and FrankaPy uses these states for message-passing. The FrankaPy PC sends control goals to the server via ROS, wrapped around standard ROS action server goals, which contain the parameters for each skill. The Robot Planner, serving as the main program, utilises the functionality provided by Franka-interface and FrankaPy. Franka-interface and FrankaPy serves as a baseline for our learning framework.

3.5 Other Software

In addition to the aforementioned software components, several Python libraries are used in this project. PyTorch is employed for various tasks including the use of Neural networks with GPU support. The subprocess library allows terminal access and is utilised to start the rosbag record command, mentioned in section 3.2. The bagpy library simplifies the loading of data from bag files. Furthermore, other libraries such as numpy, tqdm, rospy, and pickle are used for different functionalities throughout the project.

4

Robot Learning Framework

In this chapter a real-world robot learning framework for learning impedance profiles is presented. The robot learning framework is adapted from the control framework consiting of Franka-interface and FrankaPy Zhang et al. (2020).

4.1 Implementation of Impedance Controllers

Two different impedance controllers were implemented within the low level C++ framework (Franka-interface) – one with inertia shaping and another without. The implementation of the controllers satisfies Equation 2.11 with Equation 2.12, and Equation 2.14, respectively.

In the current implementation, both the desired acceleration ($\dot{\boldsymbol{v}}_d$) and velocity ($\boldsymbol{v}_d$) are set to zero. However, extending the implementation to incorporate non-zero values for these variables is possible by modifying the protobuf messages. Furthermore, the low level controller defined in Franka-interface must be updated to load and utilise these new values.

Both impedance controllers are implemented to allow for varying impedances which, just as the desired poses, are fed from FrankaPy. The impedance controllers include coriolis- and gravity force compensation, corresponding to the terms $\boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{v}_e$ and $\boldsymbol{\eta}(\boldsymbol{q})$ in the Cartesian control law. The robot is commanded with torque commands at the joint level, and the libfranka library automatically handles gravity compensation and provides access to the joint space coriolis forces. This convenient functionality makes it easier to implement coriolis and gravity compensation at the joint level, as opposed to computing these terms in task space and then compute the resulting control torques afterwards. Furthermore, the controllers are in practise not continuous, but the control commands are updated at every time interval that is given by the chosen control frequency. Then, the control torques are applied in a zero-order-hold (ZOH) fashion. This means that during the time window between two time steps, the joint motors are continuously controlled towards the last received control torques. An illustration of the ZOH principle is shown in Figure 4.1.

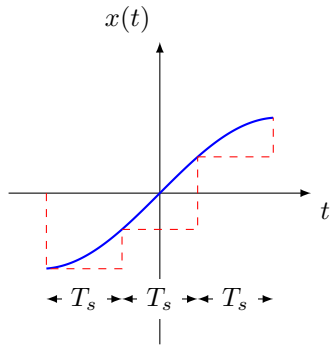


Figure 4.1: This figure shows the zero-order-hold (ZOH) principle. The blue line shows the continuous function that is being periodically evaluated. The red dotted line shows the piecewise constant function after applying ZOH to these evaluations from the continuous function.

The joint level control torques are at each time step ultimately given as

$$\boldsymbol{\tau}_c = \boldsymbol{J}^T \boldsymbol{h}_c = \boldsymbol{J}^T \left(\boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{\alpha} + \boldsymbol{h}_e\right) + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}). \tag{4.1}$$

4.2 Impedance Profile and Learning Strategy

In order to utilise the VIC capabilities, an impedance profile to feed to the impedance controller is needed. For simplicity, the rotational impedances are set to constant values, as manipulation tasks where rotation is the key movement is not considered. Examples of such manipulation task could be fastening a screw or turning a handle. The translational stiffness estimate will be on the following form

$$\hat{\boldsymbol{K}}_{P,t} = \hat{\boldsymbol{K}}_{P,t}(\boldsymbol{x}_e, \boldsymbol{v}_e, \boldsymbol{h}_e) = \begin{bmatrix} \hat{k}_x(\boldsymbol{x}_e, \boldsymbol{v}_e, \boldsymbol{h}_e) & 0 & 0 \\ 0 & \hat{k}_y(\boldsymbol{x}_e, \boldsymbol{v}_e, \boldsymbol{h}_e) & 0 \\ 0 & 0 & \hat{k}_z(\boldsymbol{x}_e, \boldsymbol{v}_e, \boldsymbol{h}_e) \end{bmatrix}, \tag{4.2}$$

where $\hat{k}_x(\cdot)$, $\hat{k}_y(\cdot)$ and $\hat{k}_z(\cdot)$ corresponds to the translational stiffness in x-, y- and z-direction, respectively. For all experiments that utilises impedance control in this thesis, the damping matrix is derived directly from the stiffness matrix to ensure a critical damped system. This holds for impedance estimates as well as impedance parameters utilised by the impedance controller during execution. Specifically the damping matrix is diagonal, and each component is computed as $d_{x,y,z} = 2\sqrt{k_{x,y,z}}$. However, if desired, the damping matrix elements can be estimated separately.

To learn such an impedance profile we suggest a learning strategy based on the learning from demonstration (LfD) and imitation learning paradigms. The proposed robot learning strategy is summarised in Algorithm 1.

First, the robot is put into what we call *guide mode*, which corresponds to applying the control law

$$\boldsymbol{\tau}_c = \boldsymbol{g}(\boldsymbol{q}).$$

Algorithm 1 Impedance Learning from Demonstration

1: Kinaesthetic Teaching Phase:
   - Record N desired end effector pose trajectories from kinaesthetic teaching demonstrations by a human operator.
   - Control the robot in pure gravity-compensation mode, enabling free movement by the operator.

2: Execution Phase with Impedance Control:
   - Execute the recorded end effector trajectories using impedance control at a desired control frequency.
   - Save end effector poses, velocities, and forces periodically during execution – typically at the same rate as the control frequency.

3: Neural Network Training:
   - Set up a supervised learning problem using a neural network.
   - Input sensory data from the execution phase (end effector poses, velocities, and forces).
   - Train the neural network to estimate impedances observed during execution.

In this mode the robot is only controlled to compensate the gravitational forces, so that the robot can be moved freely and the task at hand can be demonstrated by a human operator. While having the robot in guide mode for some period of time, the robot pose is periodically saved at a constant frequency. These robot poses then make up the desired trajectory for the task at hand. This demonstration process can be repeated an optional number of times, so that several trajectories correspond to the same task.

The next step is to publish these trajectories to the robot, one at a time, by using impedance control with a constant impedance. The data collected from the resulting robot trajectories will be the basis for a supervised learning problem. This initial run is assumed to be a satisfactory, but not perfect demonstration, and imitation learning can be used to replicate its behaviour in a VIC framework.

Finally, the goal is to learn an impedance profile for a specific robot task. Given the end effector pose, generalised velocity and external wrench, the estimator in (4.2) produces suitable impedance values for the task at hand. Using a neural network structure is a promising approach as the aim is to learn a general, but unknown relationship between two sets of values: the generalised position ($\boldsymbol{x}_e$), velocity ($\boldsymbol{v}_e$), and external wrench ($\boldsymbol{h}_e$) on one side, and the corresponding impedance values $\boldsymbol{K}_{P,t}$ on the other.

For an impedance controller without inertia shaping, there is coupling between translation and rotation in the closed-loop dynamics (see (2.15)). However, in terms of the desired control wrench $\boldsymbol{h}_c$, there is no coupling between translation and rotation, given that the stiffness and damping matrices are block diagonal. Moreover, if the impedance matrices are diagonal, k_x only affects the x-component of the control force $\boldsymbol{f}_c$, and equivalently for the other impedance values. Hence, if only the translational impedances are learned, there is no need to include the rotational terms in the loss function.

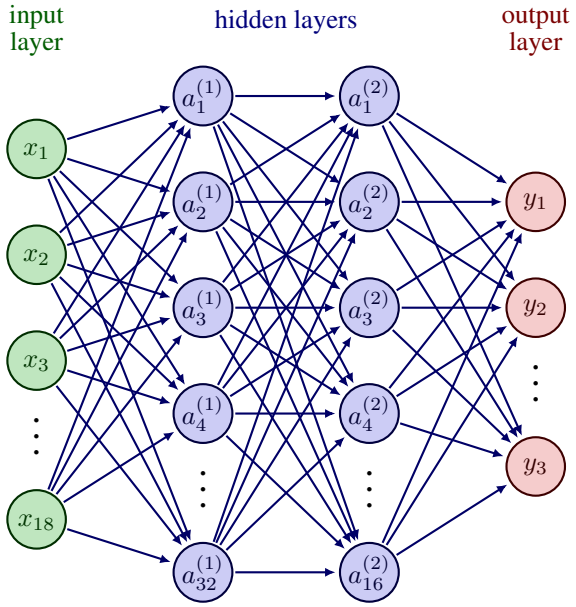


Figure 4.2: Fully Connected Neural Network with 18 inputs, 3 output, and two hidden layers with 32 and 16 neurons, respectively.

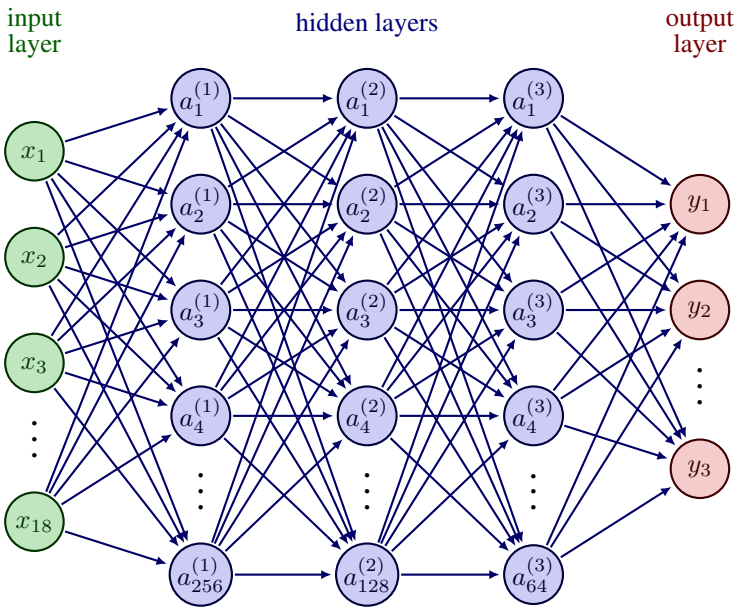


Figure 4.3: Fully Connected Neural Network Architecture with 18 inputs, 3 output, and three hidden layers with 256, 128 and 64 neurons, respectively.

The loss function used to train this neural network consists of a mean squared error (MSE) term and an L_1-regularising term, and is defined as

$$\mathcal{L} = \frac{1}{N}\sum_{t=0}^{N-1} \|\boldsymbol{f}_{c,t} - \hat{\boldsymbol{f}}_{c,t}\|^2 + \lambda_{L_1}\sum_{i,j}\|w_{i,j}\|, \tag{4.3}$$

where N is the number of time steps, $\boldsymbol{f}_c$ is the end effector control force, and $\hat{\boldsymbol{f}}_c$ is the estimated control force corresponding to the translational impedance values $\hat{\boldsymbol{K}}_{P,t}$.

If only impedance control is considered in the execution phase (step 2 of Algorithm 1), then the loss function could technically have been defined as $\mathcal{L} = \|\boldsymbol{K}_{P,t} - \hat{\boldsymbol{K}}_{P,t}\|$. However, we assume that the actual impedances $\boldsymbol{K}_{P,t}$ are unknown, to avoid learning something already known. Furthermore, there are no restricting assumptions regarding the control law used for trajectory execution. Thus, it is possible to learn an impedance profile for a type of impedance controller that is different to the controller used during trajectory execution (step 2 of Alg. 1). One example would be to use a Cartesian impedance controller without inertia shaping, and learn an impedance profile for an impedance controller with inertia shaping that best matches the demonstrated behaviour. This alternative approach, however, is not tested or analysed further in this thesis.

4.3 Setting Up the Software Environments

In this section, the process of setting up the software environments is briefly discussed, however it is not explained in detail. Furthermore, the code contributions will be showcased briefly.

To utilise the robot learning framework, two computers set up with Ubuntu 20.04 are necessary. One of the computers will run FrankaPy, while the other runs Franka-interface. The computer running FrankaPy is referred to as the FrankaPy PC while the computer running Franka-interface is referred to as the Control PC.

The Control PC needs to be set up with a real-time kernel, while the FrankaPy PC preferably has a GPU for accelerated training of neural networks. A detailed guide on how to install Franka-interface is provided in Appendix A. For the installation of FrankaPy, the reader is referred to Zhang et al. (b).

The code utilised on the two computers throughout this thesis, is published on Github, and may be viewed at `https://github.com/magneje/franka-interface` and `https://github.com/magneje/FrankaPy`. The code is adapted directly from Zhang et al. (a) and Zhang et al. (b), but also includes the contributions of this thesis.

A detailed overview of code files, indicating the contributions of this master's thesis and the preceding specialisation project, is included in Appendix B.

5

Experiments

In this chapter, the results and findings from learning the impedance profile for a robotic assembly task are presented. The assembly task to be considered is one that includes the assembly of two Duplo bricks – one mounted to a table and the other one held by the Panda robot's end effector tool. The goal of the experiments is to test the robot learning framework, and evaluate whether the framework is suitable for learning compliant and complex robotic manipulation tasks. Each step of the conducted experiments will be thoroughly explained. Finally, performance metrics used to assess the learning framework are presented. Figures and performance metrics presented in this chapter, are discussed in more detail in chapter 6.

5.1 Learning Impedance Profiles for Assembly Task

This section provides an overview of the experimental setup and examines the three steps of the learning algorithm (Alg. 1) employed in the Duplo experiment.

In order to test the learning framework data must be collected. The collection of data is done at 100 Hz, i.e. at the same rate as the control frequency. For the purpose of testing the learning framework, an assembly task is considered, where the objective is to connect two Duplo bricks. One brick is held by the robot end effector, while the other one is fixed on a table. This experimental setup, including the robot and the Duplo bricks, is shown in Figure 5.1. Furthermore, a more close-up picture of the Duplo experiment during impedance-controlled trajectory execution (step 2 of Alg. 1) is shown in Figure 5.2. Each step of the learning algorithm (Alg. 1) utilised in the Duplo experiment will be thoroughly explained in this section. The experimental parameters are summarised in Table 5.1.

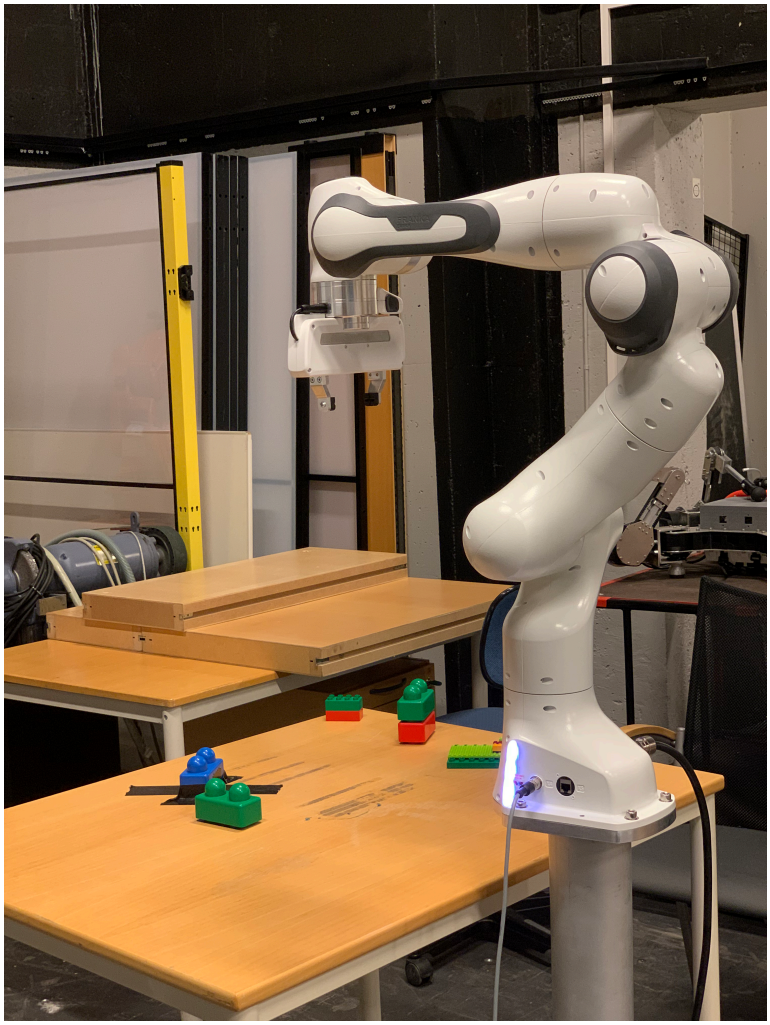


Figure 5.1: This figure shows the experimental setup of the Panda robot for the Duplo experiment.

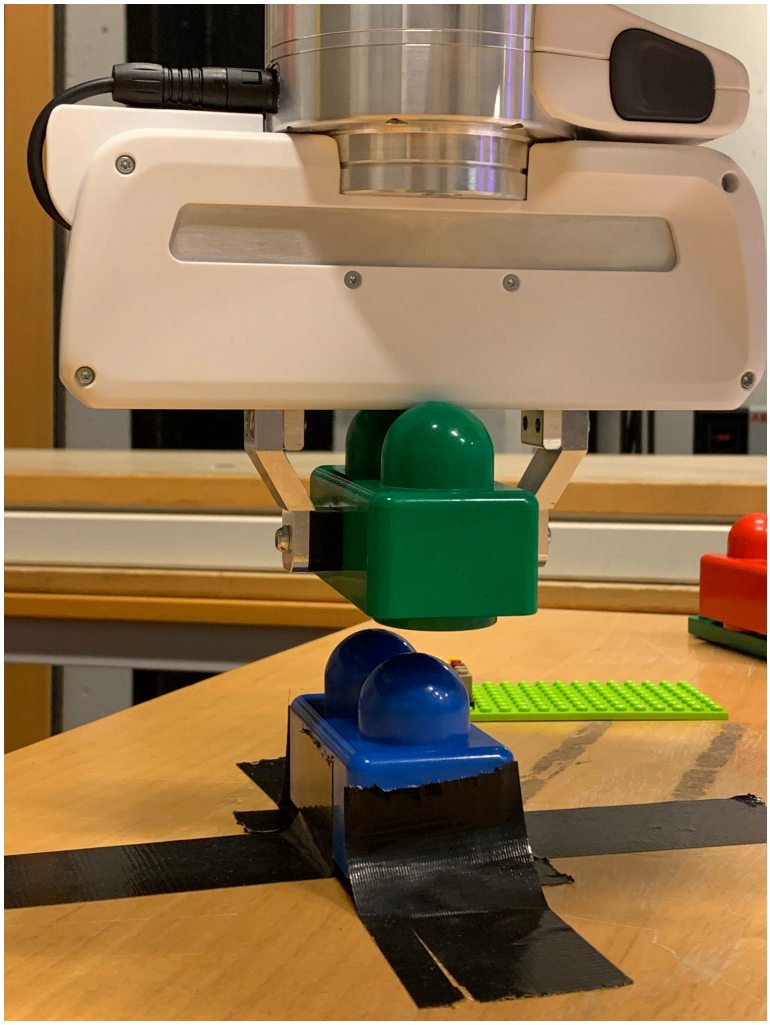


Figure 5.2: This figure shows a picture of the Panda robot during impedance-controlled trajectory execution of the Duplo experiment.

Experimental parameter	Value
Control frequency	100 [Hz]
Duplo width	0.04 [m]
Duration of demonstration	8.0 [s]
Total duration	10.0 [s]
Translational stiffness	800.0
Rotational stiffness	30.0

Table 5.1: This table summarises the experimental parameters used in the Duplo assembly experiment.

Step 1: Trajectory Recording from Demonstrations

First, data is collected from demonstrations using kinaesthetic teaching. The end effector pose is recorded at each time step for eight seconds, and these poses form the discrete-time desired trajectory. As it is hard for a human to move the robot optimally, e.g. as straight or as energy efficient as possible, the resulting desired trajectories typically became oscillatory. To achieve a smoother trajectory, post-processing filters could be applied to minimise the observed oscillations. Another approach that could improve the trajectory would be to record essential key checkpoints instead of the entire trajectory. Subsequently, a trajectory generator could be employed to interpolate between these checkpoints. The main focus of the Duplo experiment is to learn something from the data and not to generate an optimal trajectory. Thus, the desired trajectories are left without such post-processing. For the assembly task at hand, non-zero forces are required to successfully complete the brick insertion, due to friction and imprecise alignment of the bricks. Impedance control has proved unsuccessful of converging to its desired pose when the external wrench is non-zero (Villani and Schutter, 2008, p. 168). For this reason, the desired goal position is intentionally displaced 3 cm below the actual goal position. As a result, the desired trajectory is extended by two seconds compared to the eight second recorded trajectory, resulting in a planned trajectory of ten seconds. Only the z-component of the desired position is displaced, whereas the x- and y-components and the desired orientation is left unchanged. This displacement of the desired z-trajectory is observed in Figure 5.4, by looking closely at the desired z-trajectory in the final two seconds. Since the displacement is applied to make the end effector forces non-zero at the goal position, the causality between effort (force) and flow (position or velocity) is reverted compared to impedance control (Hogan, 1985a). Here, the input is the displaced target position while the output is the resulting force, which resembles admittance control.

The desired velocities are intentionally set to zero. While better tracking performance *might* be achieved by using suitable values for the desired velocity, it is not necessary for the purpose of testing the robot learning framework. Given that the assembly task at hand has a fixed goal pose, precise tracking is not essential far from this goal.

As the proposed learning algorithm (Alg. 1) suggests, a desired number of demonstrations can be performed, whereas each demonstration can have a different starting poses.

One might suspect that several demonstrations will be advantageous for learning the stiffness profile, and this statement will be investigated. For the Duplo experiment nine demonstrations will be performed, plus an extra demonstration for validation. Furthermore, the performance will be evaluated using only one, three, and all nine demonstrations, respectively. The motivation for choosing these particular number of demonstrations, is to investigate the lower limit of necessary demonstrations for effective learning, but also explore whether more demonstrations yield better performance. An upper limit is set to 10 demonstrations, as each demonstration takes *at least* 10 seconds, and much more including setup and resetting between demonstrations.

Step 2: Impedance-Controlled Trajectory Execution

Next, the desired trajectories recorded from demonstrations are executed using impedance control without inertia shaping, i.e. following the control law in Equation 2.14 and with diagonal impedance matrices. The translational stiffnesses are set to $k_x = k_y = k_z = 800$ and the rotational stiffnesses are set to $k_{o,x} = k_{o,y} = k_{o,z} = 30$. These values were chosen based on tests, high enough to perform the assembly task, but low enough to be compliant and not exert too high contact forces. Although not a necessity, the damping matrix is adapted from the stiffness by critical damping, i.e. $d = 2\sqrt{k}$. To collect data from these impedance-controlled executions, rosbag is utilised, as explained in section 3.2. This approach was chosen because fetching and saving data within the control framework proved inadequate in meeting the real-time requirements, considering the control frequency of 100 Hz. Moreover, since the proposed learning strategy involves off-line learning, meaning that the learning process occurs separately from the robot's task execution, there is no concern regarding the lack of immediate access to the data saved using rosbag. All the data will be processed simultaneously, eliminating the need for immediate access to the recorded information. Rosbag is, however, not suited for using variable impedance learning (VIL) together with VIC *online*, as data cannot be easily fetched until the rosbag recording has been stopped. Rosbag is suitable for saving data efficiently, but not for accessing new data quickly. In the bag-file, a variety of data is stored, however, only certain parts of this data is needed, namely the generalised position $\boldsymbol{x}_e$, velocity $\boldsymbol{v}_e$ and external force $\boldsymbol{h}_e$.

Plots of impedance-controlled trajectory executions of the validation trajectory, one demonstration and nine demonstrations, following the desired trajectories obtained in Step 1, is shown in Figures 5.3, 5.4, and 5.5. All figures presented in this chapter will be discussed more thoroughly in the discussion in chapter 6.

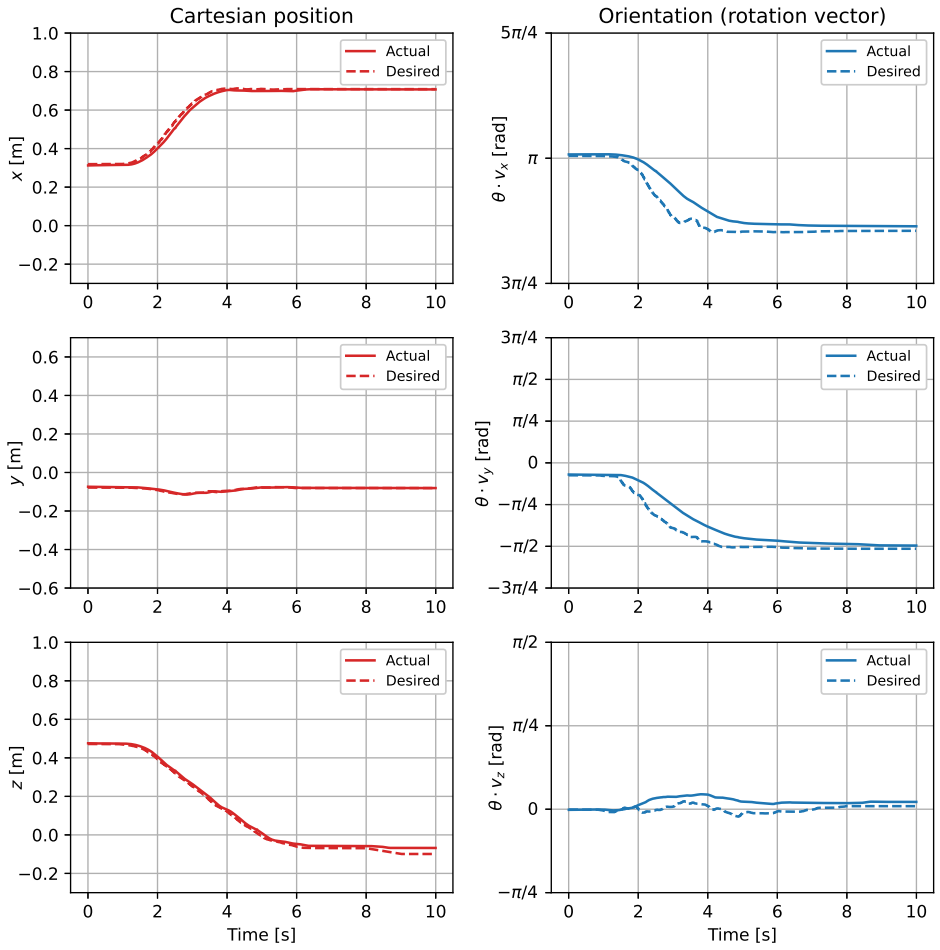


Figure 5.3: Pose validation

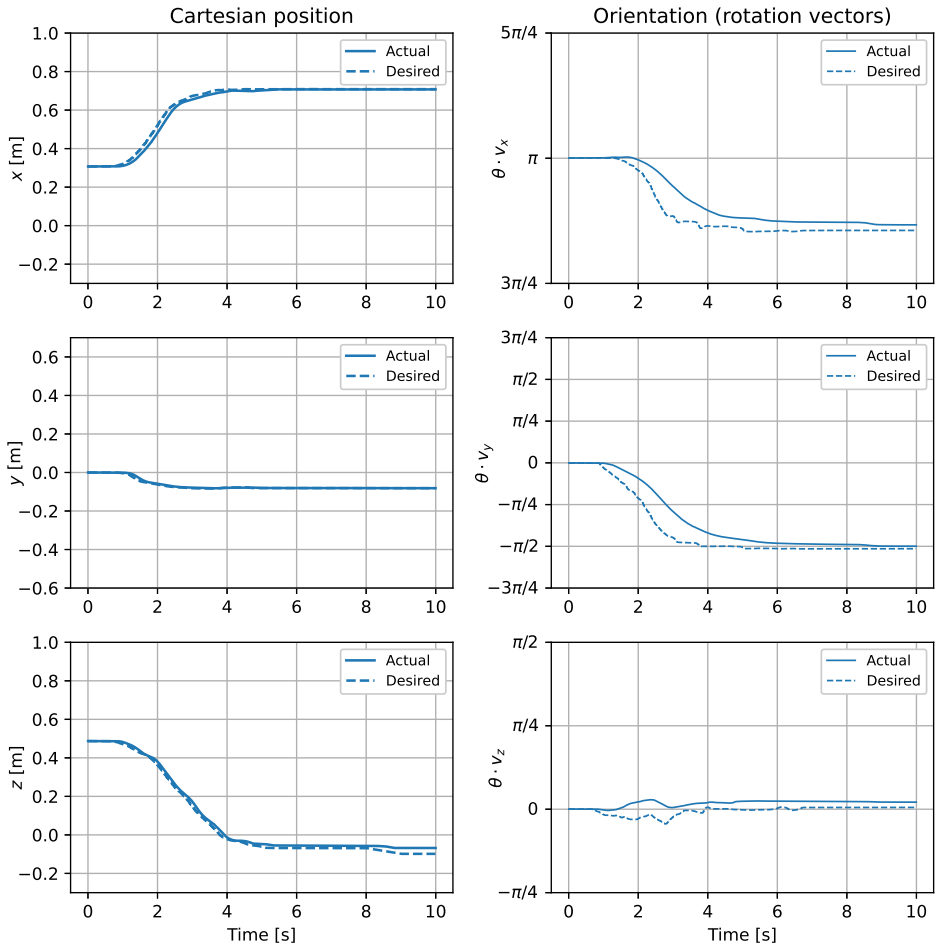


Figure 5.4: Pose, 1 demonstration

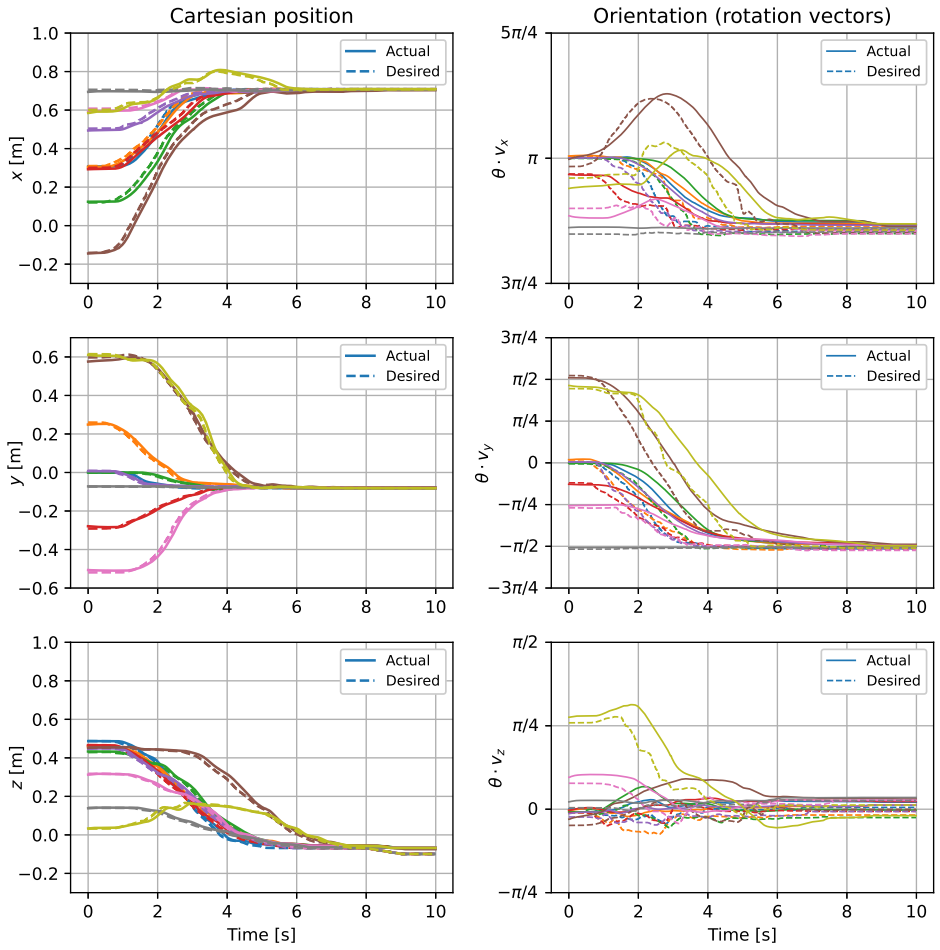


Figure 5.5: Pose, 9 demonstrations

Step 3: Learning Impedance Profile through Imitation Learning

The third step of the learning algorithm (Alg. 1) utilises the collected data to learn an impedance profile that best matches the observed behaviour. In cases where multiple demonstrations are utilised, the data is stacked in the time-direction. Specifically, the last data point from the first demonstration is followed by the first data point from the second demonstration, and so forth.

The learning procedure is repeated for two different neural network architectures, corresponding to Figure 4.2 and Figure 4.3 respectively. Furthermore, each architecture is trained with data from only one, three and all nine demonstrations, resulting in a total of six learned models. Each model is trained for 100 epochs using the Adam optimiser, and with hyperparameters as given by Table 5.2.

Plots of selected impedance and force trajectories from the experiments are shown here, while the rest are added to Appendix C for completeness. The plots shown here include impedance and force from model 1. Impedance trajectories for one demonstration are shown in 5.6 and 5.7, while impedance trajectories for nine demonstrations are shown in 5.8 and 5.9. Force trajectories for one demonstration are shown in 5.10 and 5.11.

Hyperparameter	**Model 1** (32×16)	**Model 2** ($256 \times 128 \times 64$)
Number of epochs	100	100
Learning rate	$1e{-4}$	$1e{-4}$
k_{min}	0.1	0.1
k_{max}	1500	1500
Batch size	64	64
L_1-gain (γ_{L_1})	$5e{-4}$	$5e{-5}$

Table 5.2: The hyperparameters used to train each neural network model.

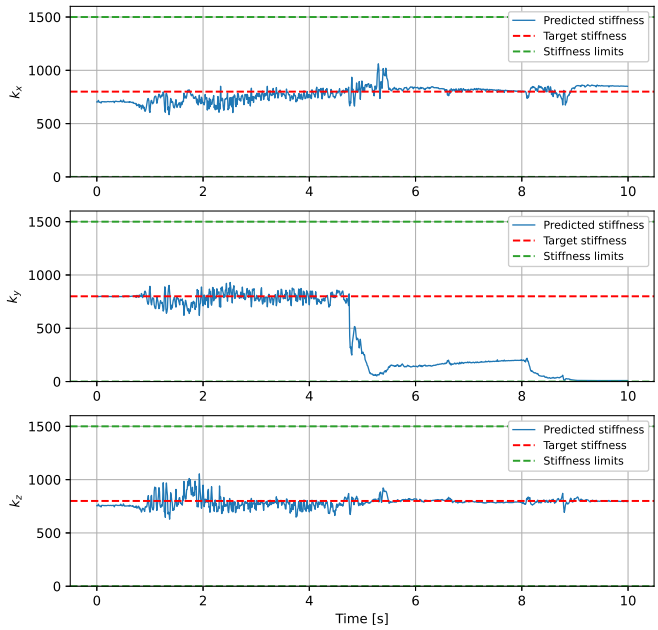


Figure 5.6: Impedance, 1 demonstration, (32×16) training

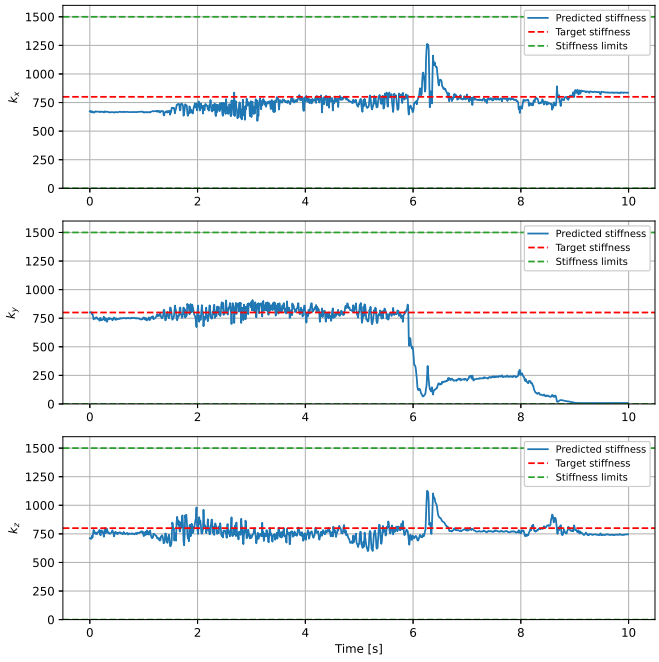


Figure 5.7: Impedance, 1 demonstration, (32×16) validation

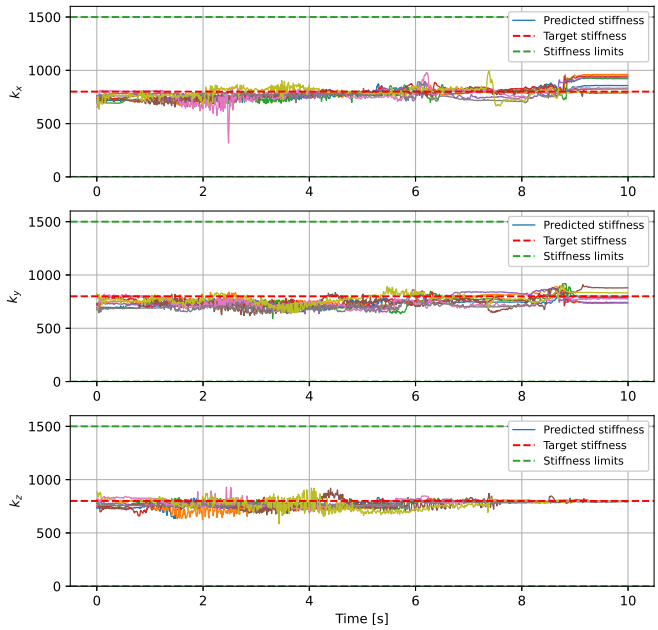


Figure 5.8: Impedance, 9 demonstrations, (32×16) training

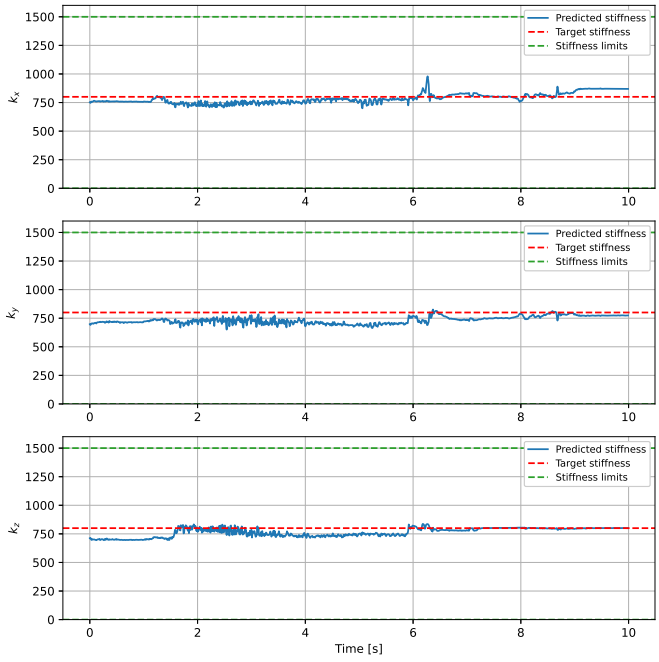


Figure 5.9: Impedance, 9 demonstrations, (32×16) validation

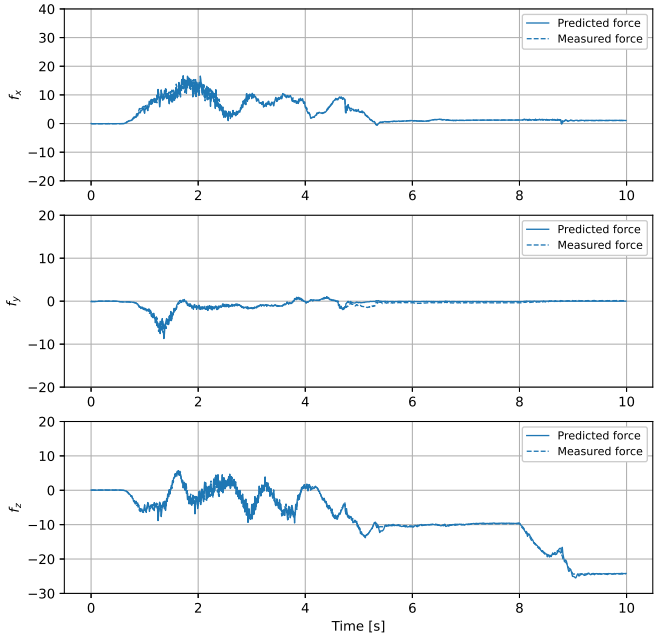


Figure 5.10: Force, 1 demonstration, (32 × 16) training

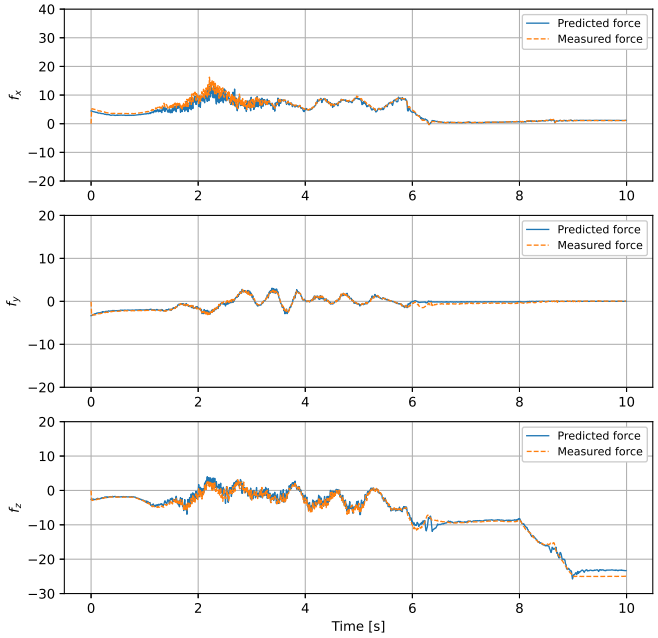


Figure 5.11: Force, 1 demonstration, (32 × 16) validation

5.2 Performance Evaluation and Metrics

This section includes metrics we will use to evaluate the performance of our neural network models, impedance estimation capabilities, and the robot learning framework overall. Results are briefly presented in this section, but are analysed and discussed thoroughly in chapter 6.

The training progress per epoch of two of the six models are included here, while the rest is included in Appendix C. In particular, the training progress of model 1 (32×16) with data from one demonstration is shown in Figure 5.12, while the training progress of model 2 ($256 \times 128 \times 64$) with data from three demonstrations is shown in Figure 5.13.

Next, the final mean squared error (MSE) loss for each model after training for 100 is shown in Table 5.3, corresponding to the final values the training progress plots (5.12 and 5.13). Note that each of the 12 values in this table corresponds to a force trajectory that is included either in this chapter or in Appendix C. Similarly, the final MSE stiffness is shown in Table 5.4. Also here, each value has a corresponding impedance trajectory that is included in this chapter or in Appendix C.

Finally, to include some evaluation metric for the real-world robot learning framework itself, the training times for each model are shown in Table 5.5.

Model / Number of demonstrations	**Model 1** (32×16) (Training / Validation)		**Model 2** ($256 \times 128 \times 64$) (Training / Validation)	
1 demo	0.57	0.62	0.34	0.47
3 demos	0.57	0.36	0.44	0.39
9 demos	0.57	0.31	0.52	0.43

Table 5.3: This table includes the final MSE loss from (4.3) for each model after training for 100 epochs, whereas both training and validation losses are shown. The values are based on the force errors, i.e., the difference between the control forces predicted from the impedance estimates, and the actual control forces.

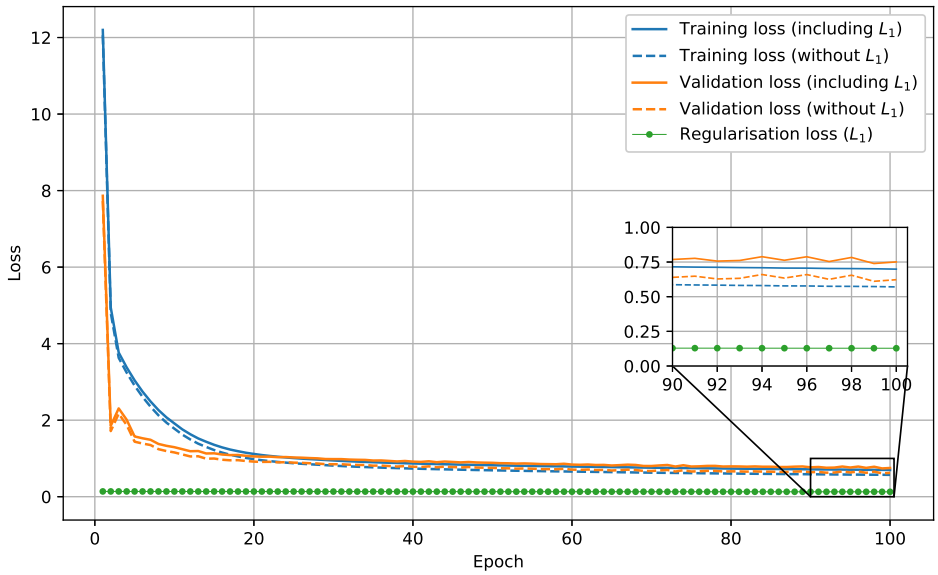


Figure 5.12: Training progress, 1 demonstration, (32×16)

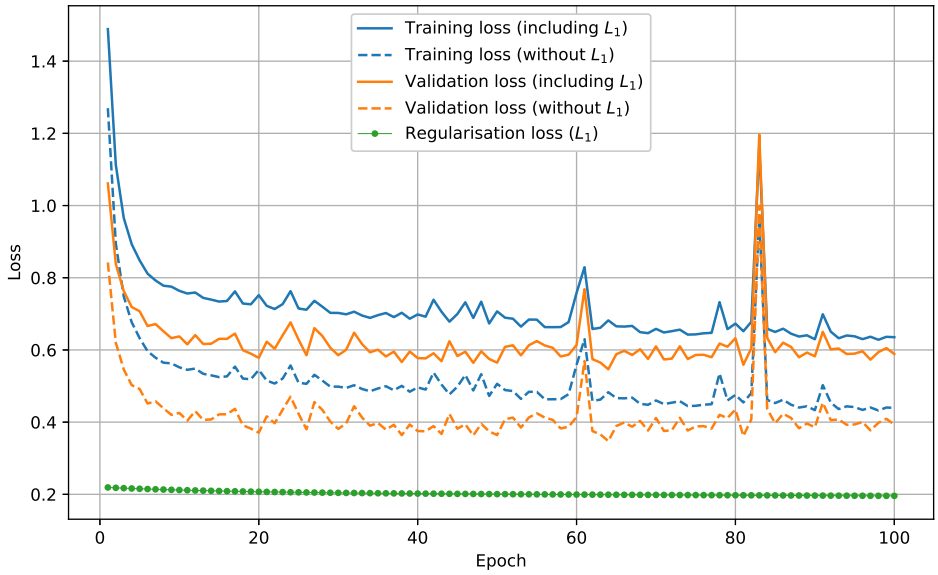


Figure 5.13: Training progress, 3 demonstrations, $(256 \times 128 \times 64)$

Model / Number of demonstrations	**Model 1** (32×16) (Training / Validation)		**Model 2** ($256 \times 128 \times 64$) (Training / Validation)	
1 demo	8.4×10^4	6.6×10^4	1.1×10^5	8.5×10^4
3 demos	4.0×10^3	3.5×10^3	4.8×10^3	5.0×10^3
9 demos	3.4×10^3	3.4×10^3	3.5×10^3	4.0×10^3

Table 5.4: This table shows the MSE stiffness for each model after training for 100 epochs, whereas both training and validation losses are shown. The stiffness error is the difference between the stiffness estimates $\hat{\boldsymbol{K}}_P$ and the true stiffness $\boldsymbol{K}_P$.

Model / Number of demonstrations	**Model 1** (32×16) (GPU / CPU)		**Model 2** ($256 \times 128 \times 64$) (GPU / CPU)	
1 demo	3.2 [s]	1.8 [s]	3.7 [s]	2.8 [s]
3 demos	9.1 [s]	5.2 [s]	10.7 [s]	7.7 [s]
9 demos	27.4 [s]	14.3 [s]	31.5 [s]	22.3 [s]

Table 5.5: This table shows 100 epoch training times for each model, both when using GPU and CPU.

6

Discussion

This chapter includes a detailed discussion on whether the overarching problem has been resolved, namely, the implementation of a real-world robot learning framework suitable for performing compliant manipulation tasks. First, the experimental results from chapter 5 are discussed, before discussing the robot learning framework itself. Finally, limitations of the work conducted in this thesis are presented, together with suggestions for future work.

6.1 Experimental Results

In this section, the experiments presented in chapter 5 will be analysed and discussed.

Neural Network Performance Evaluation

We begin by evaluating the performance of our robot learning framework in the conducted experiments. The final mean squared error (MSE) loss values from the neural network training for each model architecture (32×16 and $256 \times 128 \times 64$) and different numbers of demonstrations (1, 3, and 9) are presented in Table 5.3.

Regarding the neural network performance evaluation (see Table 5.3), we observe interesting trends. Model 2 ($256 \times 128 \times 64$) performs better with only one demonstration, while model 1 (32×16) outperforms as the number of demonstrations increases. This finding is surprising as, intuitively, one might guess that larger neural networks require more data for effective learning. However, the relationship between neural network size and performance depends on numerous factors, including regularisation terms, hyperparameters, and activation functions, and is not sufficiently understood by the research community. For model 1, the validation loss consistently decreases with an increasing number of demonstrations, while the training loss remains constant or slightly decreases. However, for model 2, the training loss increases with more demonstrations and only a slight decrease is observed in validation. We hypothesise that model 2 may tend to overfit the training data, whereas model 1 learns a more general impedance profile that leads to su-

perior results on validation data. Smaller networks like model 1 are typically less prone to overfitting, contributing to their better generalisation capabilities.

Notably, the validation loss is often lower than the training loss, which is somewhat obscure. This discrepancy could be caused by specific demonstrations being more "aggressive," involving starting poses further from the goal pose and faster movements during kinaesthetic teaching. These movements might be more prone to measurement errors and time delays, influencing the loss values.

Impedance Learning

Moving on to the impedance learning aspect, we find that impedance estimation performs worse compared to force estimation, as shown in Table 5.3 and Table 5.4. While the neural network is aided by force estimates, and better force performance is expected, the difference between the two estimations is larger than expected. This raises questions about the suitability of the used loss function and prompts consideration of other alternatives that might improve performance.

Choice of Loss Function

Next, we consider the choice of the loss function for impedance estimation. Given that force estimation yields better results, we contemplate improvements for impedance estimation. A straightforward improvement would involve including the coriolis force in the loss function, as it is included in the actual controller used during impedance controlled executions of demonstrated trajectories (step 2 of Alg. 1). The term was excluded due to its limited accessibility in the robot framework. Comparing measured joint torques with computed coriolis forces during tests indicated that the coriolis force was negligible. However, without further testing, the impact of adding coriolis force remains uncertain.

Model Architectures

We compare the results for the two different neural network model architectures (32×16 and $256 \times 128 \times 64$) based on Tables 5.3, 5.4, and 5.5. Model 1 seems to have better generalisation for the specific task, and the oscillations in the loss per epoch for model 2 (see Figure 5.13) suggest a certain level of instability. However, for more complex tasks requiring more data, model 2 could be a viable alternative, though this hypothesis is not investigated further in this thesis. Overall, both models manage to follow force trajectories, which was the primary goal aiding the models. However, the overall objective of accurately estimating impedances remains a different discussion.

Impact of Number of Demonstrations

We investigate the impact of the number of demonstrations on the learning process and model performance indicated by Table 5.3. Generally, a higher number of demonstrations positively influences performance. Going from one to three demonstrations leads to significant improvement, while further increasing to nine demonstrations yields only slight gains. Surprisingly, extending to nine demonstrations even results in worse outcomes

for model 2 compared to using three demonstrations. This discrepancy could be caused by demonstrations 4-9 being less representative, featuring more extreme movements not sufficiently validated. Adding extra validation trajectories with similar "extreme" characteristics could provide insights into this observation. However, increasing the number of demonstrations consistently gives better impedance performance, as shown in Table 5.4.

Training for one demonstration gives satisfactory force estimation performance, but the impedance estimation is underwhelming. An example of this is shown in Figure 5.6, where the y-component shows underwhelming performance compared to Figure 5.10. The problem seem to arise when a component of the end effector control force is close to zero, as the corresponding impedance value is less impactful in this case. In the case of perfect position and velocity tracking, the impedance values are not affecting the control force at all, which is clearly indicated by the control law used during the experiments (2.14).

Towards the end of each trajectory zero-velocity is expected, as the assembly task considers a goal pose at rest. Zero position error is also expected given that the Duplo brick is attached well to the table and cannot move. Duct tape was used to attach the Duplo brick to the table, and this allowed the goal Duplo brick to move slightly. Looking at Figure 5.5, slightly different goal poses are observed for each demonstration, indicating the presence of such movements. Because impedance values does not affect the control force when the position- and velocity errors are zero, a bad impedance estimate is from a controller perspective not a significant problem in these cases. However, if the impedances are quickly increased afterwards, this might cause instability (see Theorem 1[1]). Anyhow, the forces in x- and y-direction are expected to be close to zero during insertion, as the end effector is guided towards the goal pose by the goal Duplo brick, making the control force small and impedance less important.

6.2 Robot Learning Framework

Architecture of the Robot Learning Framework

The robot learning framework is based on Franka-interface and FrankaPy, which makes up the control framework shown in Figure 3.3. Our developments as compared to this baseline framework include:

- A Neural network module utilising PyTorch.
- Code for experiments utilising the proposed learning algorithm (Alg. 1)
- A new variable impedance controller in Franka-interface, supported with and without inertia shaping[2].
- New protobuf messages allowing the necessary input parameters to be fed to the variable impedance controller.

[1] Note that the theorem considers an inertia shaping controller, while the simplified control scheme without inertia shaping was used in experiments. Instability can still occur, but the specific conditions in Theorem 1 are not be valid.

[2] This controller was included in Erlandsen (2023), but has been modified slightly in this thesis. However, the inertia shaping controller showed unstable behaviour during experiments.

- Functionality for efficient data collection from experiments

The neural network module contains two classes – one for force estimation and one for impedance estimation. The force estimation is computed from the impedance estimation and introduces no extra learnable parameters. The idea is that parameters from the trained force estimator can be loaded onto the impedance estimator for inference.

Code used for the Duplo experiment, which utilises Algorithm 1, is split into two main files. The first file, "duplo_demonstrate.py", corresponds to the first two steps of the learning strategy, which concerns interaction with the physical robot. The second file, "duplo_train_model.py", corresponds to step 3 of the learning strategy, where offline learning from data is performed. The reader is referred to Appendix B for an overview of the exact files added or modified in the framework.

Training Efficiency

Training efficiency is satisfactory for the learning approaches tested in this thesis. The training times for each neural network model are shown in Table 5.5, where hyperparameters from Table 5.2 are used. Training the neural networks was primarily done utilising a GPU. In this case, the training times between the models are roughly equal. The training is only slightly slower for the larger $256 \times 128 \times 64$ model, indicating that the framework scales well regarding neural network size. PyTorch is expected to work well for training much larger and deeper networks than in this thesis, still, the relative difference in training times between the models is surprisingly small. The training times scale roughly linearly to the number of demonstrations, each demonstration adding 3 to 4 seconds, which makes sense as more demonstrations results in more training data. For completeness and performance measurements, the training procedures were repeated with CPU. Utilising CPU instead of GPU consistently gave faster training times, as shown in Table 5.5. This initially surprising result, may be caused by several factors, including the size of the neural network, the amount of data fed to the neural network, and the hardware specifications. The hardware used was a Nvidia GTX1070 GPU and an Intel Core i7-7820HK CPU, respectively. Improving the data loading procedure or optimising the batch size and other hyperparameters may be suggested for improving training performance. However, optimising the training efficiency is outside the scope of this thesis, and will not be investigated further.

Inference Time

The inference time of a neural network is the time needed for a single forward propagation, i.e., the time to make a single prediction on new data. Our neural network architecture implemented with PyTorch, proved unsuccessful in Our learned impedance profile was unsuccessful of holding a control frequency of 100 Hz when utilised online. Measurements for each neural network resulted in inference times of over $100\mathrm{ms}$, indicating that a control frequency of even 10 Hz is not possible. While it may have been possible to test lower control frequencies, doing so would have been counterproductive to the overall goal of an efficient robot learning framework capable of executing complex manipulation tasks. Executing such complex tasks requires a sufficiently high control frequency, which is why

the control frequency was set to 100 Hz. Hence, we reach the conclusion that our robot learning framework requires further improvements to be effectively utilised online.

6.3 Limitations and Future Work

While the developed robot learning framework shows promise for impedance estimation and control in certain scenarios, it also exhibits several limitations that must be acknowledged. First, the loss function used for learning impedance profiles does not perfectly correlate with the impedance performance, leading to cases where the impedance estimation is inaccurate despite a low loss value. This discrepancy could be caused by certain assumptions in the loss function, such as neglecting the coriolis effect, or other unidentified factors. Furthermore, this discrepancy was typically worse for small control forces. Future researchers are encouraged to search for a loss function that improves the correlation between force- and impedance performance.

Additionally, certain hardware limitations pose challenges to achieving optimal performance for the variable impedance controller. The absence of a force/torque end effector sensor limits the system's ability to gather essential force feedback during interactions. Although the Panda robot has torque sensors, which can be used to estimate end effector forces, a force/torque sensor is expected to give more accurate estimates. Furthermore, p. 168 in Villani and Schutter (2008) indicates that a force/torque sensor is required for impedance control with inertia shaping. This motivates future researchers to focus on improving external force/torque estimates – either by using a mounted force/torque sensor, or by improving the force/torque estimates from raw joint torque sensor data.

The framework's inability to effectively combine variable impedance control and variable impedance learning into a functional robot learning control framework represents another limitation. While each component demonstrates independent functionality, the integration of the two remains unresolved. Suggestions for future improvements include minimising the neural network inference times and possibly minimising the fetching time for reading sensor data. It is worth noting that the latter aspect requires further investigation and only applies to fetching times in FrankaPy, as fetching in Franka-interface has already proven successful.

Finally, the decision to exclusively utilise real-world robot data was established early during the research conducted for this thesis. However, the absence of simulations raises concerns regarding the generalisability of the experimental results, as the arguments against employing simulations lack supporting evidence. To reinforce the case for using real-world robot data exclusively, it would be beneficial to compare our experimental results with a simulation-based baseline or conduct a more comprehensive literature review on simulation-based learning frameworks.

7

Conclusion

This thesis aimed to develop and utilise a real-world robot learning framework, specifically a VILC framework suitable for learning compliant manipulation tasks. It also intended to explore the integration of existing machine learning approaches like RL and imitation learning into the framework. The contributions of the thesis included conducting a literature review, presenting a real-world robot learning framework, sharing experimental results from a Duplo assembly task, and proposing a novel method for variable impedance learning (VIL).

The main objectives of the thesis were partially achieved. The thesis demonstrated a robot learning framework capable of efficient communication with the robot and introduced a new approach for learning an impedance profile. However, it did not investigate the use of RL approaches, and the learned impedance profiles cannot be currently utilised online within the existing version of the learning framework. A real-world robot framework capable of VIC and VIL was demonstrated, but these approaches were not successfully combined into a VILC framework capable of learning compliant manipulation tasks.

Overall, the thesis successfully showcased advancements in the field of robotic manipulation using real-world robot data and offered valuable insights into learning-based approaches for control tasks. However, further work may be needed to fully achieve all the initial objectives outlined in the thesis.

Bibliography

Abu-Dakka, F.J., Saveriano, M., 2020. Variable impedance control and learning – a review. Frontiers in Robotics and AI 7. URL: https://www.frontiersin.org/articles/10.3389/frobt.2020.590681, doi:10.3389/frobt.2020.590681.

Anand, A.S., Abu-Dakka, F.J., Gravdahl, J.T., 2022. *Deep Model Predictive Variable Impedance Control*. arXiv preprint arXiv:2209.09614 .

Burdet, E., Osu, R., Franklin, D., Milner, T., Kawato, M., 2001. The central nervous system stabilizes unstable dynamics by learning optimal impedance. Nature 414, 446–9. doi:10.1038/35106566.

Deutche Welle, 2017. Affordable and sensitive robot panda wins prize. https://www.dw.com/en/everyman-robot-panda-wins-german-presidents-future-prize/a-41591774. [Online; accessed January 16th 2023].

Downs, L., 2003. Using quaternions to represent rotation. https://personal.utdallas.edu/~sxb027100/dock/quaternion.html. [Online; accessed; January 16th 2023].

Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T., 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning 110, 2419–2468. doi:10.1007/s10994-021-05961-4.

Erlandsen, M.J., 2023. Specialization project: "control framework for compliant robotic manipulation" .

Ferraguti, F., Secchi, C., Fantuzzi, C., 2013. A tank-based approach to impedance control with variable stiffness, in: 2013 IEEE International Conference on Robotics and Automation, pp. 4948–4953. doi:10.1109/ICRA.2013.6631284.

Franka Emika, a. libfranka. https://frankaemika.github.io/docs/libfranka.html. [Online; accessed January 16th 2023].

Franka Emika, b. Robot interface and specifications. https://frankaemika.github.io/docs/control_parameters.html. [Online; accessed January 16th 2023].

Franka Emika, 2017. Overview of franka control interface (fci). https://frankaemika.github.io/docs/overview.html. [Online; accessed January 16th 2023].

Franken, M., Stramigioli, S., Misra, S., Secchi, C., Macchelli, A., 2011. Bilateral telemanipulation with time delays: A two-layer approach combining passivity and transparency. IEEE Transactions on Robotics 27, 741–756. doi:10.1109/TRO.2011.2142430.

Gomi, H., Osu, R., 1998. Task-dependent viscoelasticity of human multijoint arm and its spatial characteristics for interaction with environments. The Journal of neuroscience : the official journal of the Society for Neuroscience 18, 8965–78. doi:10.1523/JNEUROSCI.18-21-08965.1998.

Hannaford, B., Ryu, J.H., 2002. Time-domain passivity control of haptic interfaces. IEEE Transactions on Robotics and Automation 18, 1–10. doi:10.1109/70.988969.

Hogan, N., 1985a. *Impedance Control: An Approach to Manipulation: Part I – theory*. ASME Transactions Journal of Dynamic Systems and Measurement Control B 107:1-7 .

Hogan, N., 1985b. *Impedance Control: An Approach to Manipulation: part II: implementation*. Journal of Dynamic Systems, Measurement and Control B 107:8-16 .

Kronander, K., 2015. Phd thesis: Control and learning of compliant manipulation skills. Ph.D thsis 6717.

Mahmood, A.R., Korenkevych, D., Komer, B.J., Bergstra, J., 2018a. Setting up a reinforcement learning task with a real-world robot. arXiv:1803.07067.

Mahmood, A.R., Korenkevych, D., Vasan, G., Ma, W., Bergstra, J., 2018b. Benchmarking reinforcement learning algorithms on real-world robots, in: Billard, A., Dragan, A., Peters, J., Morimoto, J. (Eds.), Proceedings of The 2nd Conference on Robot Learning, PMLR. pp. 561–591. URL: https://proceedings.mlr.press/v87/mahmood18a.html.

Ott, C., Mukherjee, R., Nakamura, Y., 2010. *Unified Impedance and Admittance Control*, in: 2010 IEEE International Conference on Robotics and Automation, pp. 554–561. doi:10.1109/ROBOT.2010.5509861.

Park, J., Choi, Y., 2020. Input-to-state stability of variable impedance control for robotic manipulator. Applied Sciences 10. URL: https://www.mdpi.com/2076-3417/10/4/1271, doi:10.3390/app10041271.

Ryu, J.H., Kwon, D.S., Hannaford, B., 2004. Stability guaranteed control: time domain passivity approach. IEEE Transactions on Control Systems Technology 12, 860–868. doi:10.1109/TCST.2004.833648.

Salisbury, J.K., 1980. *Active stiffness control of a manipulator in cartesian coordinates*, in: 1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes, pp. 95–100. doi:10.1109/CDC.1980.272026.

Slotine, J.J.E., Li, W., 1987. *On the Adaptive Control of Robot Manipulators*. The International Journal of Robotics Research 6, 49–59. URL: https://doi.org/10.1177/027836498700600303, doi:10.1177/027836498700600303, arXiv:https://doi.org/10.1177/027836498700600303.

Spyrakos-Papastavridis, E., Childs, P.R.N., Dai, J.S., 2020. Passivity preservation for variable impedance control of compliant robots. IEEE/ASME Transactions on Mechatronics 25, 2342–2353. doi:10.1109/TMECH.2019.2961478.

Stanford Artificial Intelligence Laboratory et al., . Ros wiki. https://wiki.ros.org. [Online; accessed January 16th 2023].

Stramigioli, S., Secchi, C., van der Schaft, A., Fantuzzi, C., 2005. Sampled data systems passivity and discrete port-hamiltonian systems. IEEE Transactions on Robotics 21, 574–587. doi:10.1109/TRO.2004.842330.

Sun, T., Peng, L., Cheng, L., Hou, Z.G., Pan, Y., 2019. Stability-guaranteed variable impedance control of robots based on approximate dynamic inversion. IEEE Transactions on Systems, Man, and Cybernetics: Systems 51, 4193–4200. doi:10.1109/TSMC.2019.2930582.

Villani, L., Schutter, J.D., 2008. *Force control, Chapter in Handbook of Robotics*, Springer-Verlag, pp. 161–170.

Waldron, K., Schmiedeler, J., 2008. *Kinematics, Chapter in Handbook of Robotics*. Springer-Verlag.

Yang, C., Ganesh, G., Haddadin, S., Parusel, S., Albu-Schaeffer, A., Burdet, E., 2011. Human-like adaptation of force and impedance in stable and unstable interactions. IEEE Transactions on Robotics 27, 918–930. doi:10.1109/TRO.2011.2158251.

Zhang, K., Sharma, M., Liang, J., Kroemer, O., a. Franka-interface documentation. https://iamlab-cmu.github.io/franka-interface/. [Online; accessed November 1st 2022].

Zhang, K., Sharma, M., Liang, J., Kroemer, O., b. Frankapy documentation. https://iamlab-cmu.github.io/frankapy/. [Online; accessed November 1st 2022].

Zhang, K., Sharma, M., Liang, J., Kroemer, O., 2020. A modular robotic arm control stack for research: Franka-interface and frankapy. arXiv preprint arXiv:2011.02398 .

Appendices

Appendix A

Setting Up Franka-Interface

The goal of this section is to serve as a guide for setting up Franka-interface on the Control PC from scratch. It is necessary to have a computer running Ubuntu 20.04, which is set up with a real-time kernel. The other computer should have a GPU if CUDA is to be used. These computers are referred to as the Control PC and the FrankaPy PC, respectively. To actually utilise Franka-interface, FrankaPy must be set up on a second computer with Ubuntu 20.04. However, the installation of FrankaPy will not be included in this appendix.

Setting up the environment and the network configuration consists of the following main steps:

1. Install Franka-interface on ”server PC” (Ubuntu 20.04 with real-time kernel)
2. Install FrankaPy on client PC (Ubuntu 20.04)
3. Establish a proper connection between the two, and test the framework.
4. Install extra (Python) libraries on top of FrankaPy

The installation of Franka-interface will closely follow Zhang et al. (a), which will be referenced throughout this guide. The first step is to install ROS Noetic, which provides a robust framework for developing robotic applications. This involves following the installation instructions provided by ROS for Ubuntu 20.04. Next, Google’s protobuf library is installed, and a virtual environment is made. This can be done exactly as under ”Protobuf” and ”Virtual Environment” in Zhang et al. (a). Furthermore, one should verify that a real-time kernel is correctly installed with Ubuntu. In that case, the terminal command

```
cat /sys/kernel/realtime
```

should output ”1”. In the special case that a new user has been created on a Ubuntu system with a real-time kernel already installed, all that has to be done is to execute the following command in terminal:

```
sudo usermod -a -G realtime $(whoami)
```

The proceeding instructions are adapted directly from Zhang et al. (a), the only difference being that the code is obtained from a Github-project that includes the contributions of this thesis.

1. Next, go to the 'Documents' folder. In order to include the contributions of this thesis clone the Franka-interface repository and its submodules as follows:

```
git clone --recurse-submodules https://github.com/magneje/franka-
    interface
cd franka-interface
```

2. Install libfranka and franka_ros:

```
sudo apt install ros-noetic-libfranka ros-noetic-franka-ros
```

3. Clone libfranka corresponding to robot version. For example if the firmware is 4.x:

```
bash ./bash_scripts/clone_libfranka.sh 4
```

4. Build libfranka:

```
bash ./bash_scripts/make_libfranka.sh
```

5. Build Franka-interface:

```
bash ./bash_scripts/make_franka_interface.sh
```

6. Enter the franka virtual environment and run commands:

```
source franka_virtual_env/bin/activate
pip install catkin-tools empy
bash ./bash_scripts/make_catkin.sh
```

7. Source catkin_ws:

```
source catkin_ws/devel/setup.bash
```

8. Add the following to the end of the /.bashrc file (make sure to replace "path/to" by the absolute path):

```
source /path/to/franka_virtual_env/franka/bin/activate
source /path/to/franka-interface/catkin_ws/devel/setup.bash --
    extend
```

Appendix B

Overview of Code Contributions within Franka-interface and FrankaPy

In this section, an overview of the code files that I have added or modified in Franka-interface and FrankaPy is presented. Some of these changes were made in the specialisation project (Erlandsen, 2023). As a result Table B.1 and Table B.2 below, are extensions of similar tables from Section 4.1 of Erlandsen (2023). All code including these modifications is available online at `https://github.com/magneje/franka-interface` (which based on Zhang et al. (a)) and `https://github.com/magneje/FrankaPy` (which is based on Zhang et al. (b)).

Folder location	Filename
Franka-interface:	
franka-interface/include/franka-interface/feedback_controller/	cartesian_variable_impedance_feedback_controller.h
franka-interface/src/feedback_controller/	cartesian_variable_impedance_feedback_controller.cpp
FrankaPy:	
examples/	run_vic.py
examples/	duplo_demonstration.py
examples/	duplo_train_model.py
examples/	duplo_plot.py
examples/	test_popen.py
examples/	test_pytorch.py
examples/	test_pytorch_inference.py
examples/	test_rosbag.py
frankapy/	nn.py

Table B.1: Files added to the Franka-interface-FrankaPy framework.

Folder location	Filename
Franka-Interface:	
franka-interface-common/include/franka-interface-common/	definitions.h
franka-interface/	CMakeLists.txt
franka-interface/proto/	feedback_controller_params_msg.proto
franka-interface/proto/	sensor_msg.proto
franka-interface/src/	feedback_controller_factory.cpp
franka-interface/src/termination_handler/	termination_handler.cpp
FrankaPy:	
bash_scripts/	start_control_pc.sh
frankapy/	franka_arm.py
frankapy/	franka_constants.py
frankapy/	franka_interface_common_definitions.py
frankapy/proto/	feedback_controller_params_msg.proto
frankapy/proto/	sensor_msg.proto
frankapy/	skill_list.py
frankapy/	utils.py

Table B.2: Files modified in the Franka-interface-FrankaPy framework.

Appendix C

Extra Figures from Experiments

This section includes figures not included in chapter 5, where experimental results from a Duplo experiment was presented. In particular, the figures that will be presented include end effector trajectories, stiffness estimates, force estimates and neural network training progress for different number of demonstrations and network architectures No new models or concepts are introduced in this section, and the reader is referred to chapter 5 for details regarding the experiments.

This appendix includes all figures that are basis for the numerical results from Table 5.4 and Table 5.3, and that are are included in chapter 5. Furthermore, it includes pose trajectories from three demonstrations, and training progress of the models not included in chapter 5.

First, the pose trajectories form three demonstrations are shown in Figure C.1.

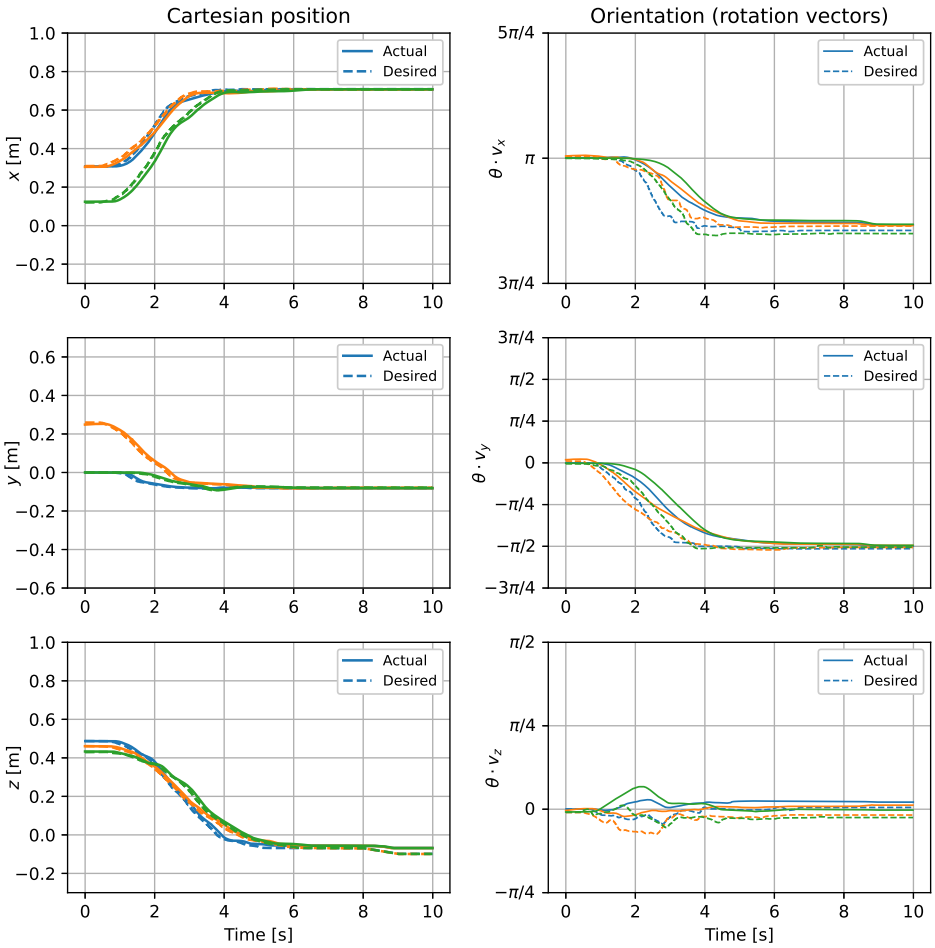


Figure C.1: Pose, 3 demonstrations

Impedance Estimates

The following section includes the impedance estimates for each model that are not included in chapter 5.

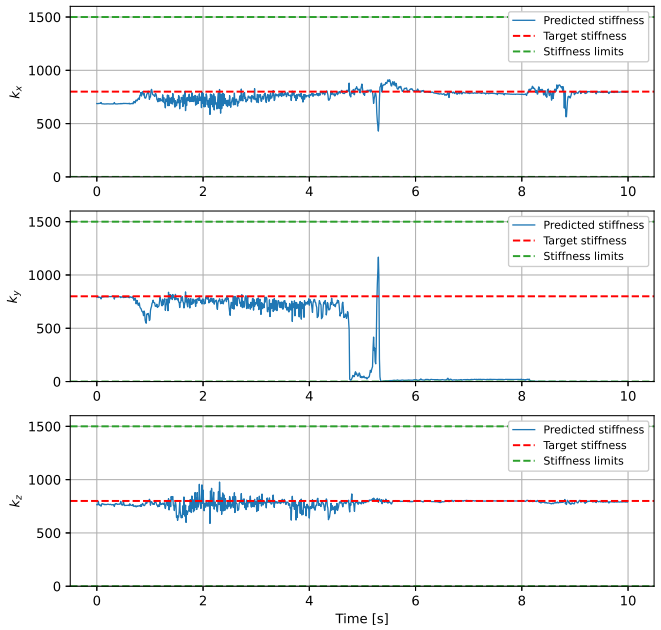


Figure C.2: Impedance, 1 demonstration, ($256 \times 128 \times 64$) training

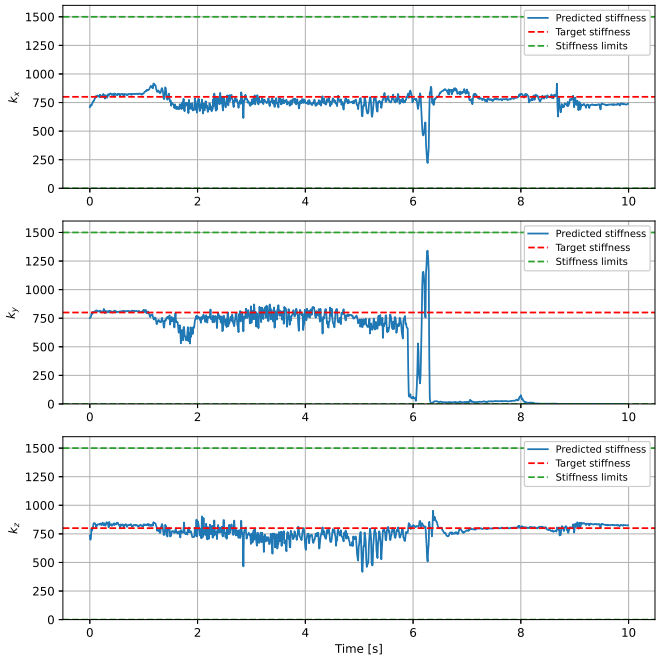


Figure C.3: Impedance, 1 demonstration, ($256 \times 128 \times 64$) validation

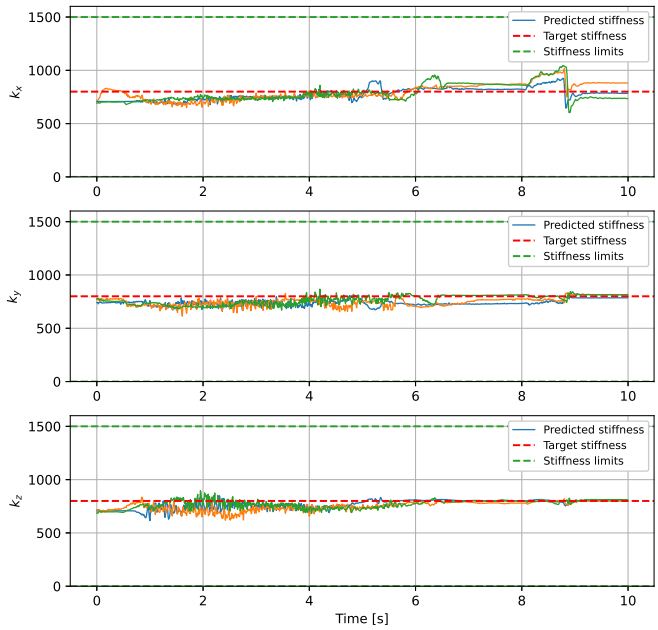


Figure C.4: Impedance, 3 demonstrations, (32×16) training

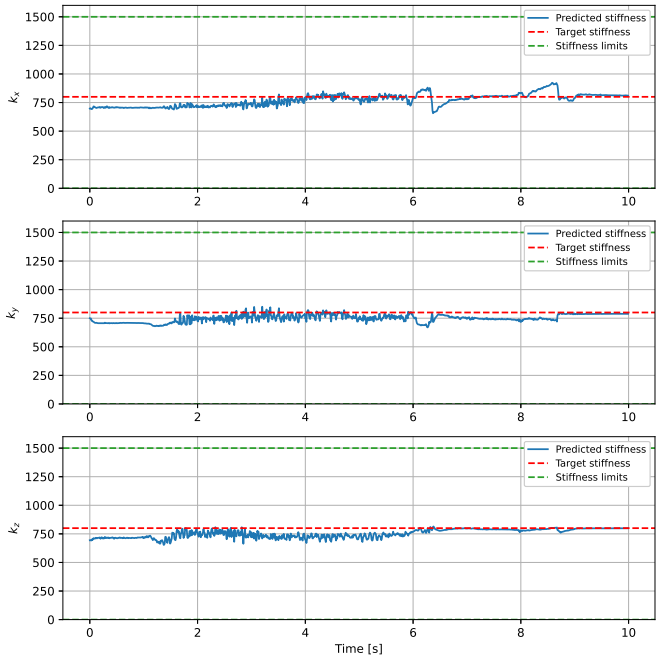


Figure C.5: Impedance, 3 demonstrations, (32×16) validation

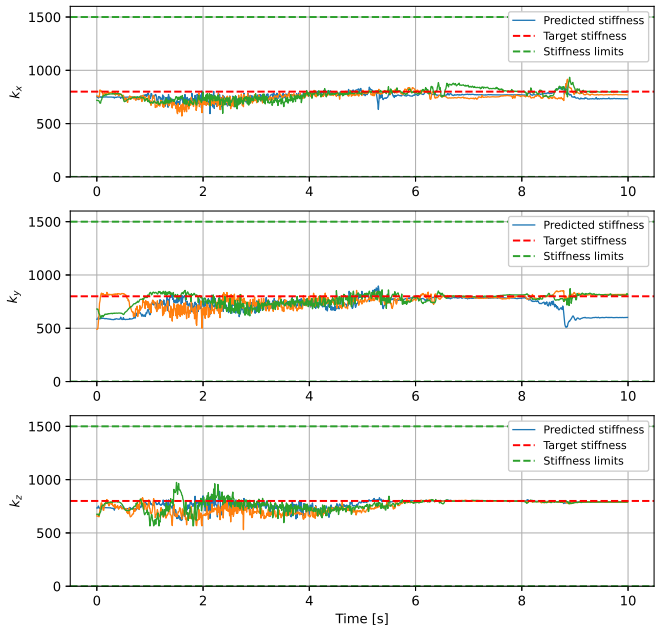


Figure C.6: Impedance, 3 demonstrations, $(256 \times 128 \times 64)$ training

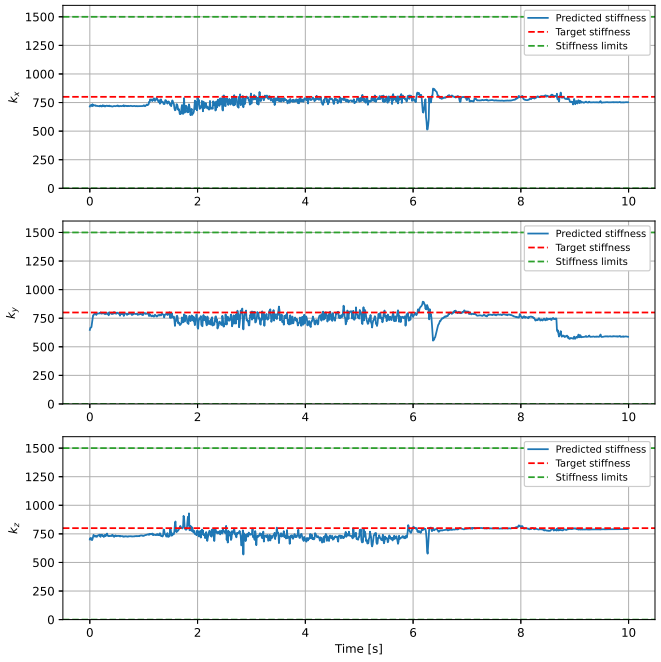


Figure C.7: Impedance, 3 demonstrations, $(256 \times 128 \times 64)$ validation

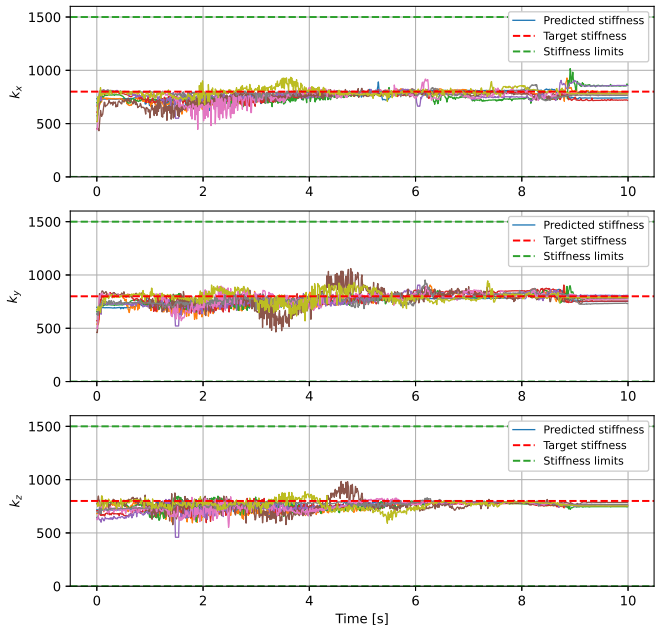


Figure C.8: Impedance, 9 demonstrations, $(256 \times 128 \times 64)$ training

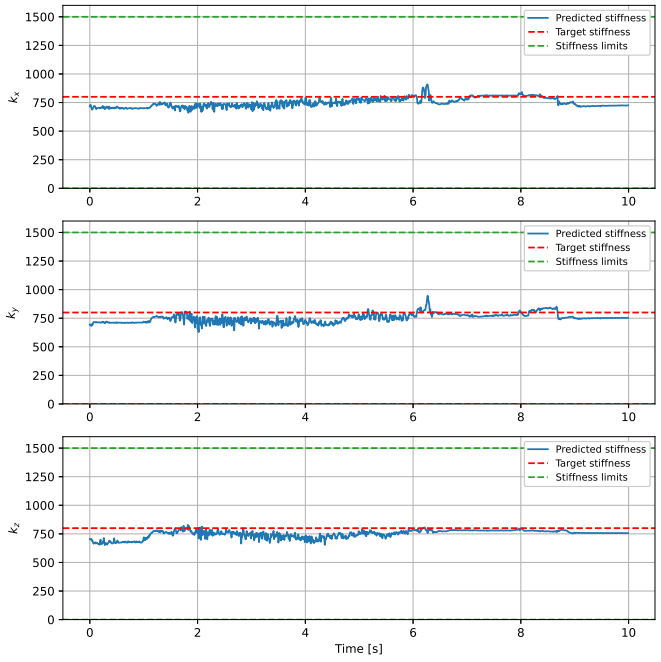


Figure C.9: Impedance, 9 demonstrations, $(256 \times 128 \times 64)$ validation

Force Estimates

The following section includes the force estimates for each model that are not included in chapter 5.

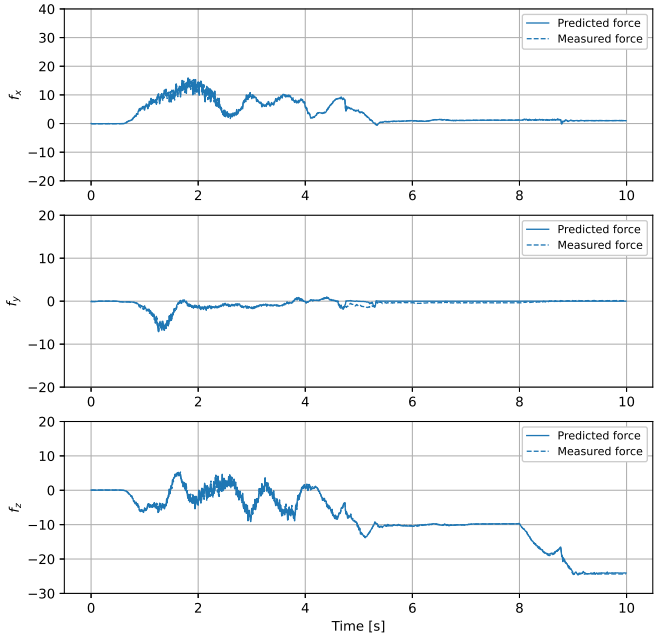


Figure C.10: Force, 1 demonstration, $(256 \times 128 \times 64)$ training

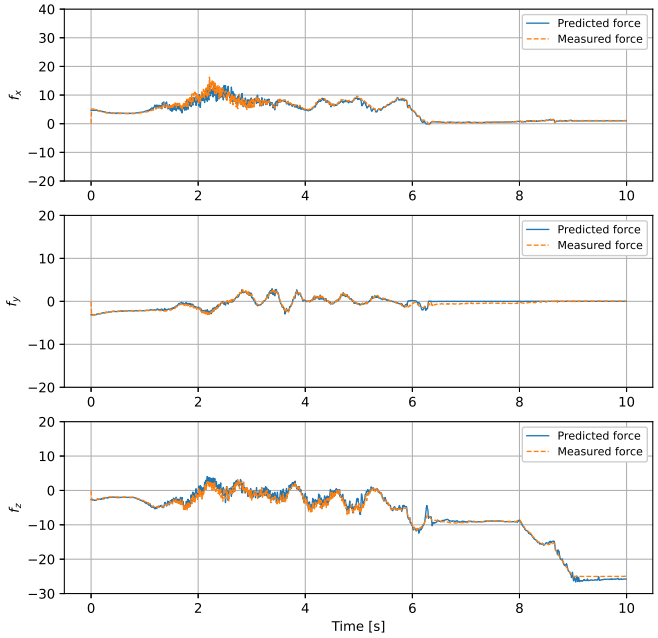


Figure C.11: Force, 1 demonstration, $(256 \times 128 \times 64)$ validation

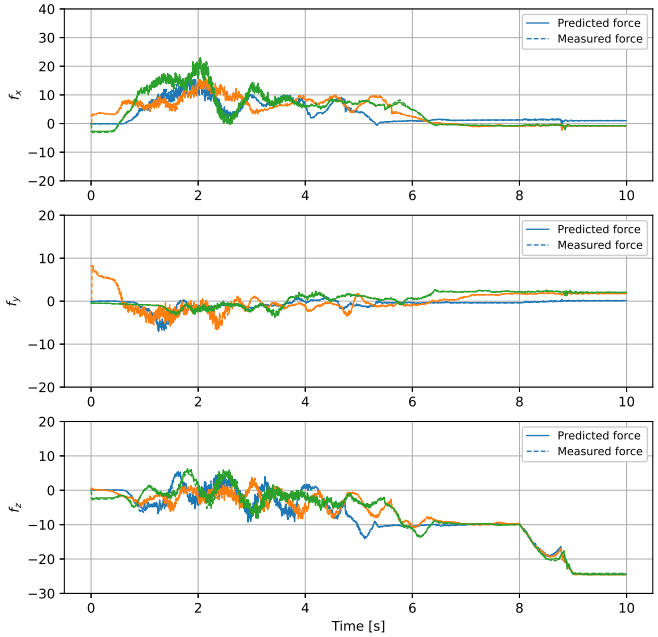


Figure C.12: Force, 3 demonstrations, (32 × 16) training

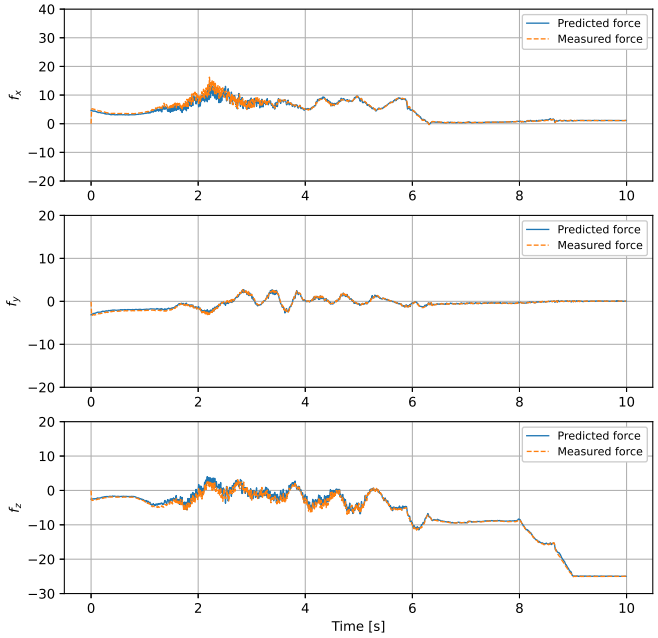


Figure C.13: Force, 3 demonstrations, (32 × 16) validation

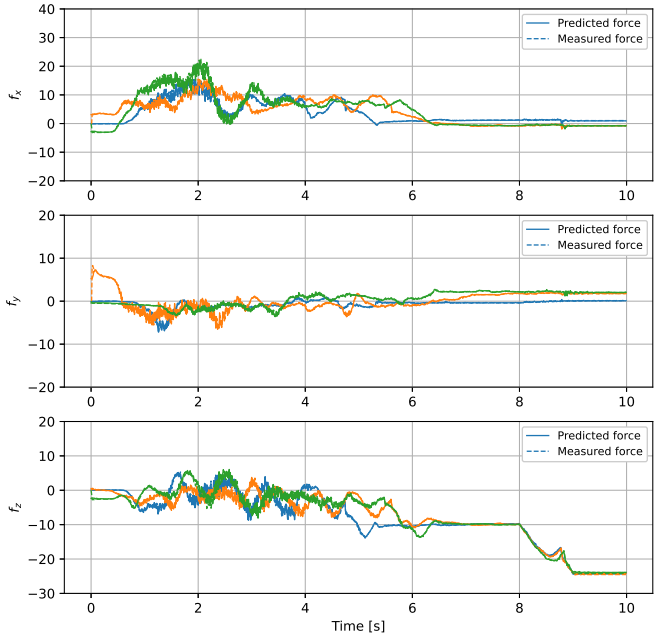


Figure C.14: Force, 3 demonstrations, $(256 \times 128 \times 64)$ training

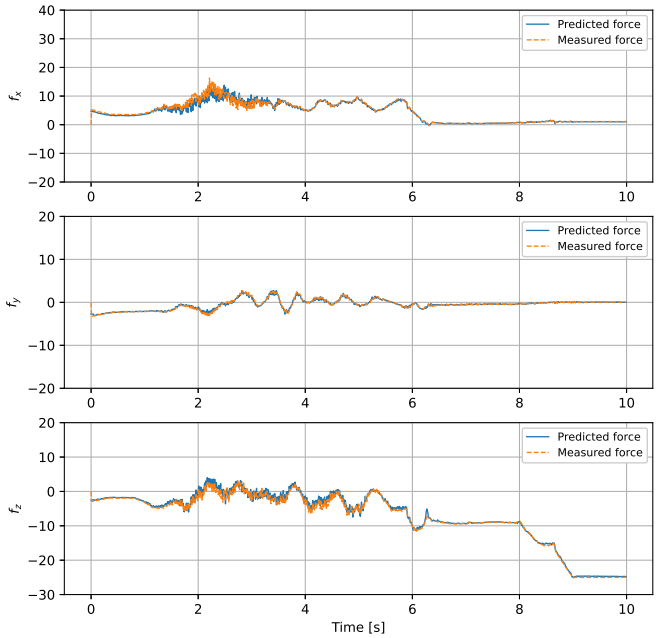


Figure C.15: Force, 3 demonstrations, $(256 \times 128 \times 64)$ validation

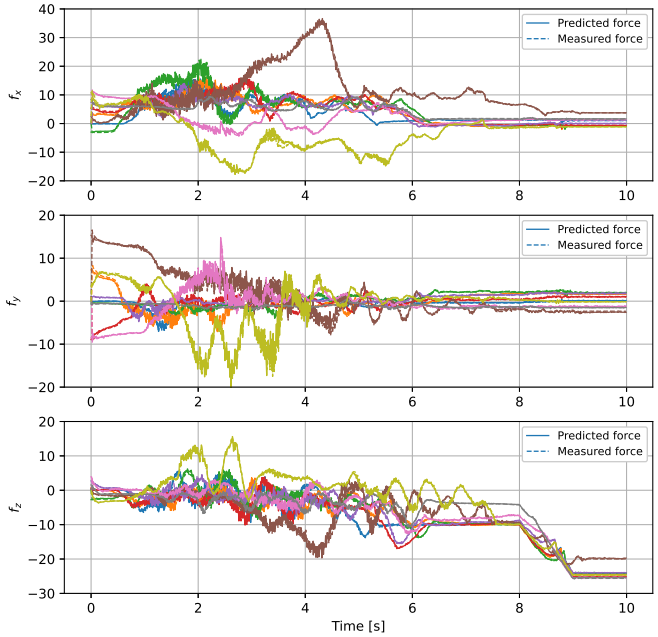


Figure C.16: Force, 9 demonstrations, (32 × 16) training

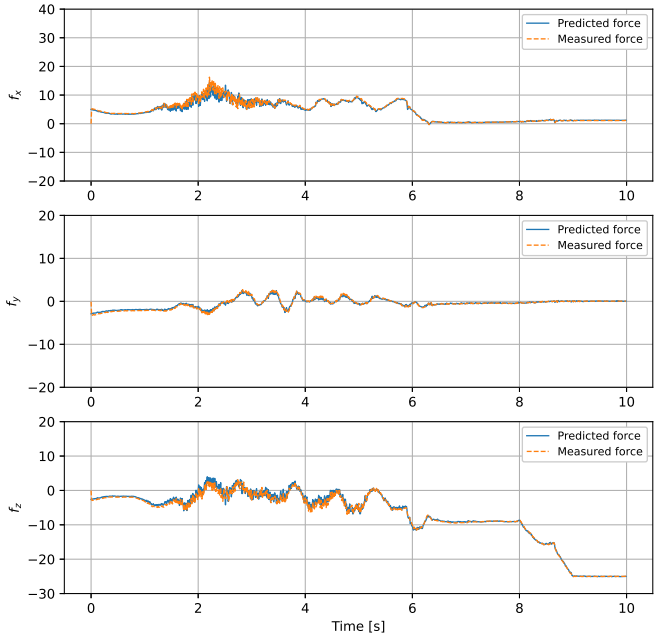


Figure C.17: Force, 9 demonstrations, (32 × 16) validation

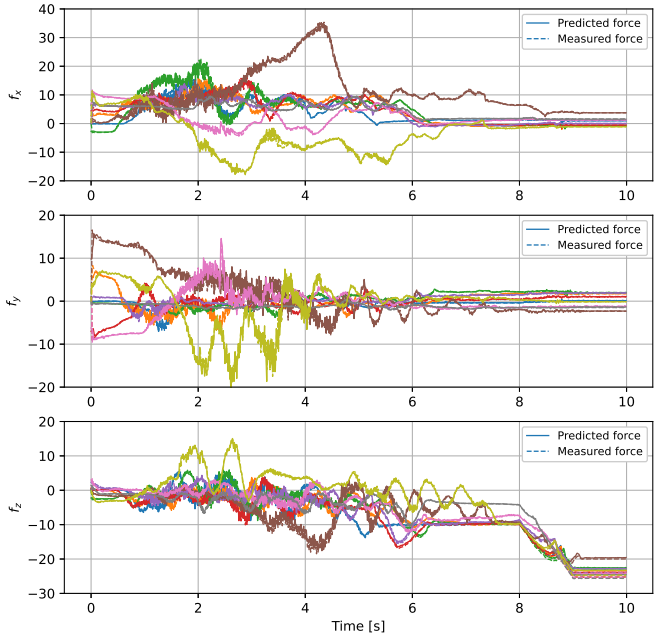


Figure C.18: Force, 9 demonstrations, $(256 \times 128 \times 64)$ training

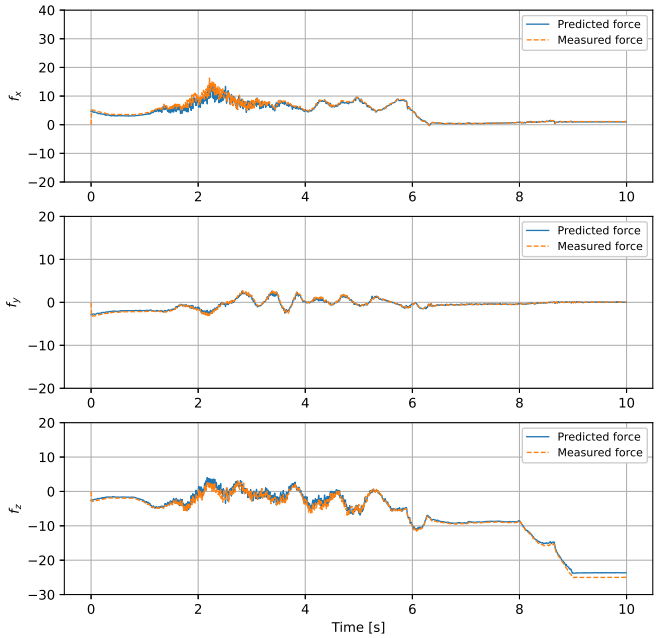


Figure C.19: Force, 9 demonstrations, $(256 \times 128 \times 64)$ validation

Training Progress of Neural Network Models

The following section includes figures of each model's training progress per epoch. Only results from the models not included in chapter 5 are included.

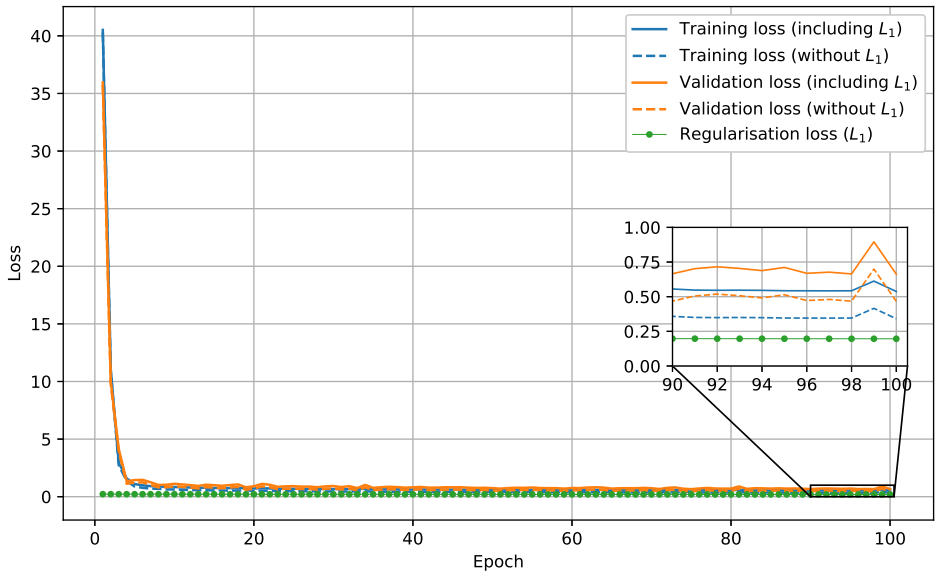


Figure C.20: Training progress, 1 demonstration, ($256 \times 128 \times 64$)

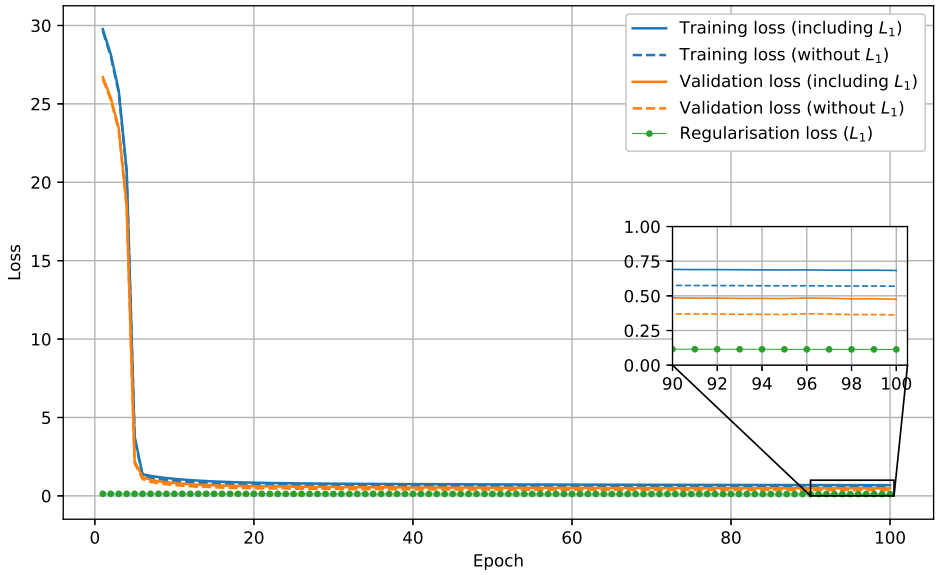


Figure C.21: Training progress, 3 demonstrations, (32×16)

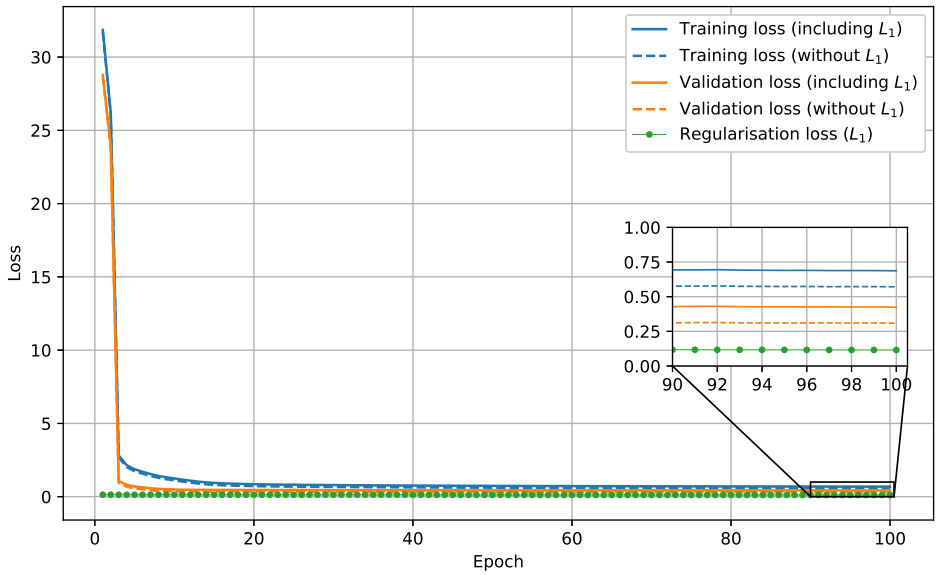


Figure C.22: Training progress, 9 demonstrations, (32×16)

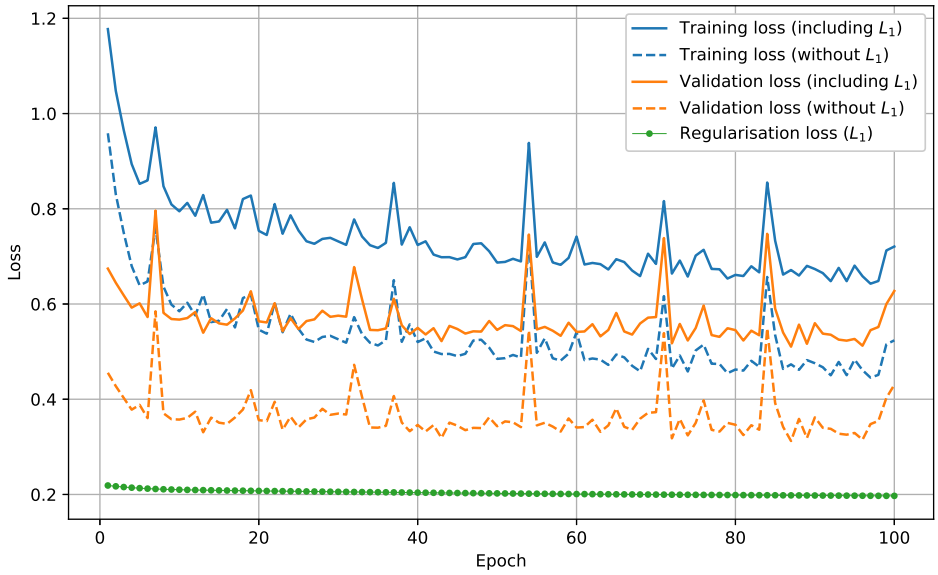
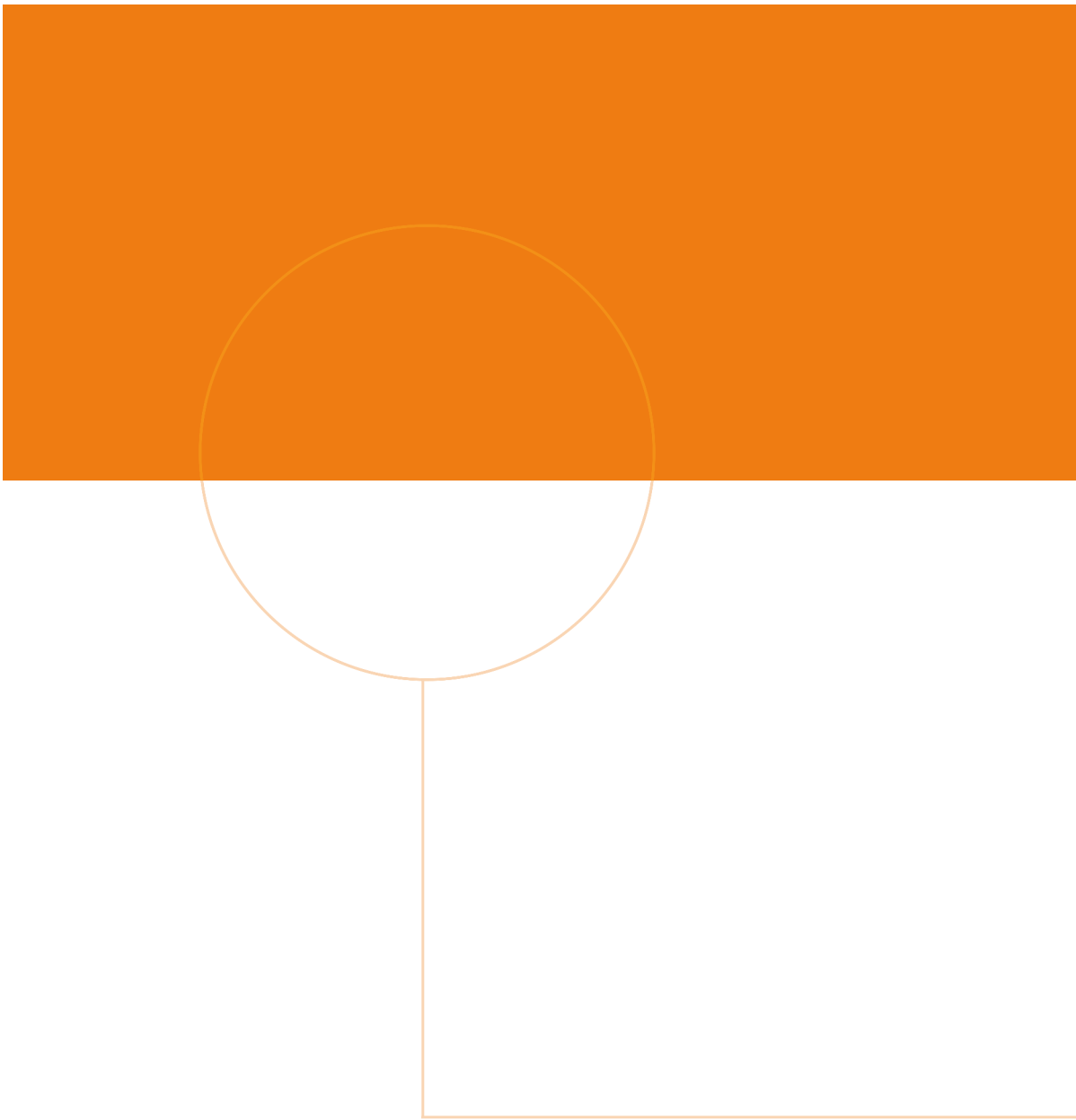


Figure C.23: Training progress, 9 demonstration, $(256 \times 128 \times 64)$



NTNU

Norwegian University of
Science and Technology