INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAE025

Thèse de doctorat

Learning and designing shared control skills from demonstrations for assistive robots

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École nationale supérieure de techniques avancées

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Signal, Images, Automatique et Robotique

Thèse présentée et soutenue à Palaiseau, le 16/12/2024, par

Gabriel Quere

Composition du Jury :

Paolo Robuffo Giordano Directeur de recherche, Université de Rennes (Inria/IRISA)	Président
Nathanaël Jarrassé Chargé de recherche, Sorbonne Université (ISIR)	Rapporteur
Stefanos Nikolaidis Associate professor, University of Southern California (ICAROS)	Rapporteur *Absent lors de la soutenance*
Ayse Kucukyilmaz Assistant Professor, University of Nottingham (CHART)	Examinateur
Adriana Tapus Professeur, ENSTA Paris (U2IS)	Examinateur
David Filliat Professeur, ENSTA Paris (U2IS)	Directeur de thèse
Freek Stulp Head of Department, DLR (RMC-KRO)	Invité
João Silvério Group leader, DLR (RMC-KRO)	Invité

Dedicated to my family

Remerciements

This thesis is based on my PhD research conducted at the Institute of Robotics and Mechatronics of the German Aerospace Center in Munich from 2017 to 2024. Throughout this journey, I was fortunate to be surrounded by remarkable people and learned so much.

I am deeply grateful to Dr. Freek Stulp, who gave me the opportunity to contribute to such a meaningful project. His enthusiasm, strategic vision, and attention to detail were instrumental in helping me formalize concepts and make steady progress in my work. I also want to express my thanks to Prof. Alin Albu-Schäffer for his unwavering interest and support of the EDAN project.

I am thankful to Prof. David Filliat for his guidance and for providing the freedom to explore innovative solutions for assistance on EDAN. I would also like to extend my gratitude to Dr. João Silvério for always making time to assist me. I'll remember fondly our many productive sessions, brainstorming probabilistic methods, iterating ideas and filling up whiteboards. My thanks to Dr. Franz Steinmetz for his help, especially in refactoring the Shared Control Templates repository. I am also grateful to Prof. Daniel Leidner for his early guidance in my work and for leading us to that delightful restaurant in Yokohama with undulating food. I thank my jury members for the thoughtful feedback and insightful questions.

My deepest appreciation goes to the EDAN team – past and present members – who made these years truly worthwhile: Jörn, Annette, Maged, Samuel, Jianxiang, Sebi, Miriam, Felix, Ulrike, Elle, Tai, and all our students. You all made coming to work each day something to look forward to. We shared many long lab sessions, always with a positive spirit supported by snacks, and achieved great successes together, with many more to come. I'm also thankful to the ISL group and to Flo, Werner, Max and Antonin for their invaluable support with hardware and software. A special thank you to Dr. Laura Herlant and the members of the Personal Robotics Lab, who sparked my interest in assistive robotics.

Thanks to all my colleagues and friends with whom my life is woven. Special mention to the bouldering crew – Martin, Posi, Susi, Caspar, Sebi, Dave, Antonin, Matthias, Margherita –, the DLR crew – Marie, Blanca, Laura, Lukas, Adrian, Anna, Kathy, Anton, Ria –, the dancing crew – Vero, Veronika, Clara, Iro, Joy, Lydia – and those who live far away: Aude, Guillaume, Clem, Antoine, Isabel, Teddy, Mélanie, Noémie, Alexia, Rahaf, Thibault, Simon, Lucille. Miscellaneous thanks to: Audrey, for seeding my heart with joy. Alison, for the laughs. Florence, for the template :p Pauline, for the jumps in a half-frozen lake. Inès, for those sparse but wonderful moments. Juliane, for being there. Ellie, for the poetic dreams. Lisl, for the unexpected. Noelia, for being colorful. Teresa, for your support.

Finally, I am profoundly grateful to my family for always believing in me and standing by my side. Thank you.

Abstract

Assistive robots, among which wheelchair-mounted robotic arms, hold great potential in supporting individuals with limited physical abilities by helping them interact with their environment. These robots can enhance users' autonomy by assisting with daily activities such as eating, drinking, or opening doors. However, controlling such robotic arms through accessible interfaces (like joysticks, sip-and-puff devices or buttons) can be challenging, as these interfaces are lower dimensional than the control space of the robot. Although autonomous task execution is an active area of research, involving humans in the control loop increases users' agency, leverages their situational awareness and improves the robustness of the system. Therefore, to operate these systems more easily, the development of intuitive controls and effective user interfaces is essential.

This thesis introduces a new framework called "Shared Control Templates" which provide task-specific assistance through shared control. It explores methods for designing shared control skills that can reliably assist users in completing tasks successfully, whilst ensuring ease of use and user control of key motions. For instance, the user will control the quantity of liquid that is to be poured when preparing a drink, or the appropriate opening when pulling out a drawer. To achieve this, task-specific skills – acting in task space – are represented as finite-state machines, with transitions triggered by factors such as distances, wrench generated from environmental contact, or user triggers. These skills comprise of two key components: input mappings and active constraints. Input mappings define how user commands translate into robot motions, while active constraints enforce geometric limits on the robot's end-effector task space, guiding the user and maintaining safe operation. For example, when pouring liquid from a bottle, the robot ensures no spilling by controlling the bottle's position and partial orientation, while the user determines how much liquid is poured by controlling the tilt angle.

To facilitate the design of such shared control skills, this research explores semi-automatic learning of these skills from demonstrated end-effector trajectories. Geometric shapes in Euclidean space are used as constraints in tasks such as opening drawers or cabinet doors. A library of pre-existing skills is then leveraged to accelerate the design of new skills by the skill designer. A probabilistic model – Kernelized Movement Primitives – is investigated to enable the derivation of input mappings and active constraints. This model additionally allows the adaptation of skills based on user input, enhancing both the design and execution phases.

This shared control method is integrated with an assistive robot with a world model, user interfaces and a whole-body coordination of the entire system. It enables able-bodied and motor-impaired people to accomplish sequences of activities of daily living, in various settings such as a DLR-RMC laboratory, participants homes or the 2023 CYBATHLON Challenges and 2024 CYBATHLON.

Résumé

De nombreuses personnes sont atteintes de paraplégie, ce qui peut rendre difficile ou impossible des tâches quotidiennes essentielles telles que manger, boire ou ouvrir des portes. Les robots d'assistance, tels qu'un bras robotique monté sur un fauteuil roulant, présentent un grand potentiel pour aider des personnes handicapées moteur à interagir avec leur environnement, renforçant leur autonomie. Toutefois, le contrôle de ces bras robotisés par des interfaces classiques – telle qu'un joystick, "sip-and-puff" ou boutons – peut s'avérer difficile, car elles ont peu de degrés de liberté comparé à l'espace de contrôle du robot. L'exécution autonome de tâches est un domaine de recherche actif et demande peu d'effort de la part de l'utilisateur. Cependant, impliquer des utilisateurs dans la boucle de contrôle augmente leur autonomie et tire parti de leur compréhension de la situation, ce qui au final améliore la robustesse du système. De plus, les utilisateurs apprécient en général être en contrôle du robot, et non uniquement un récipient passif d'aide. Par conséquent, pour faciliter l'utilisation de ces systèmes, il est essentiel de développer une assistance intuitive couplée à des interfaces utilisateur efficaces.

Cette thèse introduit une nouvelle méthode appelée "Shared Control Templates", qui fournit une assistance avec du contrôle partagé pour des tâches quotidiennes. Elle explore des méthodes de conception de compétences de contrôle partagé qui peuvent aider de manière fiable, transparente et personnalisée des utilisateurs à effectuer leurs tâches, leur garantir une facilité d'utilisation et le contrôle des mouvements clés. Par exemple, une utilisatrice peut ainsi contrôler la quantité de liquide versé lorsqu'elle prépare une boisson, ou l'amplitude de l'ouverture lors de la manipulation d'un tiroir. Pour ce faire, les compétences spécifiques à la tâche sont représentées comme des machines à états finis, avec des transitions déclenchées par des éléments tels que des distances, des forces générées par des contacts avec l'environnement ou des pressions de boutons. Ces compétences comprennent deux éléments clés : les correspondances de commandes utilisateur et les contraintes actives. Les correspondances de commandes utilisateur précisent comment les commandes de l'utilisateur se traduisent en mouvements du robot, tandis que les contraintes actives imposent des limites géométriques à l'espace disponible pour l'outil du robot, ce qui sert à guider l'utilisateur et assurer la sécurité des opérations. Par exemple, lorsque l'utilisateur déverse le liquide d'une bouteille, le robot s'assure de ne pas en mettre à côté en contrôlant la position et l'orientation partielle de la bouteille, tandis que l'utilisateur détermine la quantité de liquide versée en contrôlant l'inclinaison de la bouteille. L'assistance est définie par rapport à l'objet manipulé et indépendante du profile de vitesse des commandes de l'utilisateur, ce qui en facilite la modélisation par ce dernier.

Pour faciliter la conception de ces compétences de contrôle partagé, cette recherche explore l'apprentissage semi-automatique de ces compétences à partir de l'enregistrement de trajectoires du robot lors de nouvelles tâches. Des formes géométriques de l'espace Euclidien sont extraites des trajectoires par de l'optimisation sans gradient. Elles sont ensuite utilisées comme contraintes dans des compétences comme assister l'ouverture de tiroirs. Une bibliothèque de compétences préexistantes est ensuite exploitée par le designer de compétences pour accélérer la conception de nouvelles compétences. Un modèle probabiliste plus expressif - Kernelized Movement Primitives - est ensuite étudié, permettant de dériver des correspondances de commandes utilisateur et des contraintes actives. Ces contraintes forment un Generalized Cylinder, plus ou moins contraignant suivant les trajectoires démontrées. Ce modèle permet également à l'utilisateur

d'adapter l'assistance en fonction de l'environnement avec l'interface utilisée pour contrôler le robot, ce qui facilite les phases de conception et d'exécution.

Cette méthode de contrôle partagé est intégrée au robot d'assistance EDAN de l'équipe Re-enabling Robotics du DLR-RMC. EDAN est doté d'un système de perception capable de créer un modèle de l'environnement, d'interfaces utilisateur (joystick et électromyographie) et d'une coordination de l'ensemble du système, permettant de contrôler en même temps le bras robotique et le fauteuil roulant. Grâce à la méthode d'assistance proposée dans ce travail, des personnes valides et handicapées ont pu accomplir des séquences de tâches de la vie quotidienne, dans divers contextes tels que un laboratoire du DLR-RMC, le domicile des participants ou le CYBATHLON Challenges 2023 and le CYBATHLON 2024.

Contents

List of Figures

List of Tables

Glossary

AC	Active Constraint
ARAT	Action Research Arm Test
BCI	Brain Computer Interface
DLR	Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center)
DoF	Degree of Freedom
EDAN	EMG-controlled Daily AssistaNt
EEG	Electroencephalogram
EMG	Electromyography
Etasl	Expression-based Task Specification Language
GMM	Gaussian Mixture Models
GP	Gaussian Process
GUI	Graphical User Interface
IM	Input Mapping
IMU	Inertial Measurement Unit
iTaSC	Iterative Task Specification and Control
KMP	Kernelized Movement Primitives
ODB	Object DataBase
PDDL	Planning Domain Definition Language
ProMP	Probabilistic Movement Primitives
RL	Reinforcement Learning
RMC	Robotic and Mechatronic Center
SCT	Shared Control Templates
TSR	Task-Space Region
WBC	Whole-Body Control
WSR	World State Representation

Chapter 1

Introduction

Stroke, spinal cord injuries, and neuromuscular diseases frequently lead to permanent motor impairments, resulting in long-term disability. Globally, over 15 million people are living with spinal cord injury. Between 250 000 and 500 000 people are getting afflicted every year, with driving accidents the leading cause, according to the World Health Organisation [Org]. A large-scale study done in 2013 in the United States [Arm+16] reported approximately 5.4 million people affected by paralysis, of which the majority (72%) was younger than 65 years. Performing tasks of daily living can become challenging – even impossible – with motor impairment, which may lead to a dependence on caregivers for everyday life. Assistive technologies, referred to by Howard *et al.* [How+22] as "any product either specially designed and produced or generally available, whose primary purpose is to maintain or improve an individual's functioning and independence and thereby promote their well-being", re-enable those affected by motor impairments to perform activities of daily living independently. Examples of such activities include reaching for objects, drinking, eating, scratching, or opening doors [Hol+05; Che+13; Pet+22]. More than a simple tool, those devices impact how users interact within society and with their social circle and allow them to live more independently.

1.1 Assistive devices

For people with limited functionality in the upper extremities, devices such as orthoses, prosthetic hands, arms, or robotic manipulators re-enable physical interaction with the environment, see Fig. 1.1. Various interfaces have been developed to control those devices, typically measuring biosignals or body motions [ORR19]. The benefits of using assistive technology include enhancing function and independence, improved safety, and promoting social inclusion [How+22]. However, there are many challenges to their adoption, such as cost, low reliability, poor correspondence between device utility and target users' needs, lack of user involvement in the design and decision-making process, poor usability, lack of training, complex maintenance or cleaning, poor customer support and stigmatizing esthetics [ORR19; How+22]. Due to those challenges, abandonment rates are considerably high, with some studies finding their range between 20% and 70% depending on the context. Therefore, to encourage adoption, assistive devices should be easy to use, reliable, and personalized, to cater to users' needs and preferences.

1.2 Wheelchair-mounted robotic manipulators

A well-known example of assistive devices is power wheelchairs, which provide mobility and independence for those who can no longer walk. For people with paralysis, a power wheelchair on which is mounted a robotic manipulator provides mobility and the ability to physically interact with the environment [KB15].

Figure 1.1: **Assistive robots. Left**: The commercially available Stretch 3 from Hello Robots. **Center**: The EDAN robot, with which this thesis contribution is validated. **Right**: The commercially available OBI robot from Desin.

Compared to power wheelchairs with two Degrees of Freedom (DoFs), robotic manipulators have several DoFs – at least six for the end-effector pose plus one for the tool – and are, therefore, significantly more challenging to control. Additionally, the more severe a person's motor impairment, the more limited the control interfaces available to them to operate their assistive technology, with control signals getting lower in dimensionality and bandwidth. Thus, paradoxically, a greater need for sophisticated assistive devices is paired with a diminishing ability to control their additional complexity. Assistive robot arms on the market mostly use the 2D wheelchair joystick interface and are restricted to direct control [Nie+16]. Controlling all available DoFs of the manipulator with such a low-dimensional interface is achieved with 'mode switching', in which the user selects different subsets of manipulator DoFs to be directly controlled. Moreover, target users may no longer be able to operate a mechanical interface such as a joystick and might, therefore, require alternative interfaces, such as those based on bio-signals or body motions.

While users have demonstrated that the direct control of robotic manipulators re-enables them to perform challenging tasks [Mah+11], the switching between different control modes is inefficient because it breaks the flow of motion. Therefore, complex tasks requiring the combination and coordination of different DoFs of both wheelchair and manipulator, such as opening and going through a door, are incredibly challenging with direct control using low-DoF interfaces and lead to a high cognitive workload. This becomes even more crucial when controlling assistive devices with interfaces based on bio-signals, which often results in noisier commands and low throughput. To overcome those limitations, the system must provide some level of assistance to the user when doing activities of daily living.

1.3 Robot assistance

Robots can assist humans in completing tasks. Different levels of assistance can be provided, depending on the available interface and user preferences. Users should control key decisions, whether abstract (e. g. selecting an object to pick) or detailed (e. g. adjusting rotational velocity when pouring water). The coordination of the remaining DoFs – of the wheelchair, arm, and tool – can then be delegated to the system. Multiple levels of assistance have been proposed in the literature, such as in section 43.3 of [SKK08], which outlines three distinct levels of assistance.

The first level of assistance is **shared control**, which enables both the user and the robotic system to influence the manipulator's actions in real time. The user provides continuous inputs, while the robot applies constraints or corrections to ensure safe and efficient execution. For example, for a *pick mug* task, the user could decide on the approach velocity and direction of the approach towards the mug. At the same time, the assistance could constrain the end-effector height and orientation such that tilting the mug over is impossible. The division of control can change dynamically based on the context.

A second level of assistance is **shared autonomy**, where the user specify high-level tasks (e. g. selecting an object to grasp), while the robot autonomously determines and executes the required motion. By reducing the need for continuous user input, shared autonomy lowers cognitive and physical effort. For instance, for a *pick mug and drink* task, the user could select which mug to pick via a voice command or a graphical interface, let the robot autonomously pick the mug, then input a "bring me the mug for drinking" command so that the robot brings the mug to their mouth.

A third level of assistance is **full autonomy**, a system where the robot operates independently, with little to no human input required for task execution. The user only selects a goal, such as *clean the room* or *prepare a meal*. The robot should then autonomously plan and execute tasks and subtasks, as well as adapting to unforeseen effects, such as a failed grasp or an object rolling over a table. For instance, the user would ask: 'Bring me a glass of cold water,' and the robot would open a fridge to grab a glass, close the fridge, and then bring the glass to the user. While full autonomy has been demonstrated in laboratory settings for tasks like feeding [Gal+19], cleaning [Lei19], and cooking [Bee+11], real-world deployment remains challenging due to environmental variability and human unpredictability.

To provide those different modes of assistance, an assistive robot needs a comprehensive world model of its environment, appropriate interfaces for human-robot interactions, and the ability to assist users in achieving their goals. This requires control algorithms that maintain stability and reliability while coordinating the many robot DoFs, even with inaccurate control signals. An ideal assistive robot should provide this spectrum of autonomy levels [Che+13] and allow users to effortlessly switch between these levels based on their preferences and the specific task at hand.

While higher autonomy reduces the cognitive and physical demands on the user, this only holds if the robot operates reliably, executing tasks safely without causing unintended consequences. Although advances in autonomous systems in real environments such as warehouses are notable, autonomous behavior is an active area of research for unstructured environments such as homes. Robot errors, such as spilling water or breaking a mug, can be difficult or even impossible for target users to correct, leading to frustration and negatively impacting their quality of life. Involving the user in the control loop increases users' agency, leverages their situational awareness, and improves the system's reliability, which is crucial for real-world adoption. Moreover, some studies show that users often prefer maintaining control over robotic systems [Bha+20], even when it increases their workload [Kim+12]. Finally, citing Henry Evans, a quadriplegic expert user of assistive robots who participated in numerous studies: "It's very important to me, from a sense of self-worth, to do things for myself independently whenever I want, even if it is slower." For these reasons, this work focuses on shared control, balancing user agency with robotic assistance, providing support and empowerment.

1.4 Problem statement and contribution

The general objective of this thesis is to develop a shared control framework that facilitates the adoption of user-centered robotic assistive systems. It shall provide people with motor impairments with safe and easy-to-use technology that enables them to perform activities of daily living independently and effectively. Therefore, we consider the following problem:

- How to provide shared control assistance that enhances users' agency, behaves transparently, is reliable, and personalizable ?

User **agency** refers to the ability of users to make choices and exert control over their interactions with robotic systems. It encompasses how users can influence the robot's behavior, tailor its functions to meet their needs, and decide how the system assists them in various tasks. User agency is crucial for promoting independence, enhancing user satisfaction, and ensuring that the assistive technology aligns with the user's preferences and goals.

Transparency is defined by Alonso *et al.* [AD18] as: "the observability and predictability of the system behavior, the understanding of what the system is doing, why, and what it will do next". Dragan *et al.* [DLS13] further distinguishes between predictability and legibility: predictability involves forecasting the robot's trajectory, while legibility refers to the user's understanding of the robot's actions or intentions. Transparent and legible assistance fosters trust and improves usability, whereas inconsistent behavior can lead to frustration and mistrust.

Reliability: Reliable assistance means that the robot will support the user in successfully performing their desired tasks safely and consistently without malfunctions or unintended effects on the environment. Achieving reliability in autonomous robots remains a significant challenge, especially within unstructured human environments. A robot's world model is, by nature, an approximation of its real environment, with factors like friction often only roughly estimated. Moreover, errors in robotics can be costly, as user safety is paramount. Robots should not diminish the user's sense of control, particularly when the robots fail, as they sometimes will. In such cases, the robot must remain safe and offer the user the ability to recover. Preventing unintended outcomes, such as knocking over objects, is especially critical for users with motor impairments, for whom cleaning up after such mistakes can be especially challenging.

Personalization is also crucial; the robot should cater to the user's needs and preferences, which may change over time. Multiple aspects of the assistance are concerned, such as the interface used. Another aspect is the level of autonomy of the assistance behavior. For example, users may prefer precise control for some tasks while opting for higher-level goal selection in others.

Towards this objective, our contributions are:

- The definition of a generic shared control framework, Shared Control Templates (SCT), which provides a variety of consistent assistive behaviors obtained by tailored mapping of user inputs and constraining the end-effector while ensuring the user maintains control.
- Two distinct methods for learning task constraints and facilitating skill design. The first approach involves learning a multi-model representation from task trajectory demonstrations and demonstrating the ability to transfer knowledge from existing SCTs to new ones. The second approach employs a probabilistic model that applies constraints to the end-effector and allows for user adaptation to accommodate changes in environmental conditions.
- The validation of this framework through user studies with the 'EMG-controlled Daily AssistaNt' (EDAN), an assistive robot developed by the Re-Enabling Robotics team of the Robotic and Mechatronic Center of the German Aerospace Center (DLR-RMC) [Vog+20b]. Results showed successful completion of various sequences of activities of daily living, such as pouring, drinking, and manipulating articulated objects like doors, drawers, and refrigerators.

Those contributions have been presented in the following publications:

- "Shared control templates for assistive robotics", ICRA 2020 [Que+20]
- "Learning and Interactive Design of Shared Control Templates", IROS 2021 [Que+21]
- "A probabilistic approach for learning and adapting shared control skills with the human in the loop", ICRA 2024 [Que+24]

1.5 Outline

We begin by reviewing the state of the art in relevant fields in Chapter 2. In Chapter 3 is presented the assistive robot EDAN. This wheelchair-mounted manipulator possesses a world model with localized instances of known objects, two possible 3 DoFs interfaces – a joystick or an EMG-based interface – and a whole-body Cartesian impedance controller that guarantees safety for the user when interacting with the environment. In Chapter 4, we detail a novel assistance framework we developed – Shared Control Templates (SCT) – which aims to provide transparent, reliable, and personalizable assistance. Assistive skills are defined as a finite-state machine, where each state can specify the mapping of user commands and enforce task space constraints on the end-effector to guarantee safety, guide the user, and control leftover DoFs. Chapter 5 covers the results from different user experiments with able-bodied and motor-impaired users and from our participation in the 2023 CYBATHLON Challenges. We show that employing the SCT framework can be used for sequences of activities of daily living.

Then, we turn to learning from demonstrations to simplify the skill design and to provide personalized assistance based on user preferences – they might want a skill to behave in a certain way – or needs, such as a specific task they would like to be assisted with. Chapter 6 covers learning parameterized geometric constraints and bootstrapping skill design with a library of already known skills. Chapter 7 introduces a new probabilistic approach using Kernelized Movement Primitives to learn input mapping and active constraints, which users can then adapt if needed.

Finally, we conclude with Chapter 8, analyzing the advantages and limitations of our methods and further avenues for research.

Chapter 2

State of the art

2.1 Devices for assistive technology

2.1.1 Fully integrated assistive robots for people with motor impairments

Research on robotic arms as a manipulation aid dates back to the 1970s [CLS79]. The first commercially available systems, such as the MANUS manipulator, appeared in the 1990s [Kwe+89]. When integrated with power wheelchairs, robotic manipulators empower individuals with severe physical disabilities to interact with their surroundings and retain independence. Available wheelchair-mounted robotic manipulators include the iARM [RSP05] and the JACO arm [Mah+11]. These manipulators are usually controlled with the same 2D joystick used to control the wheelchair itself [DCC10; Mah+11].

The works above only provide direct control; from a control perspective, the wheelchair and the arm are independent entities. However, the robotic manipulator should not merely be attached to the wheelchair; the wheelchair's DoFs should also be available in the arm's low-level control scheme. This concept, known as whole-body control (WBC), has proven beneficial in humanoid robots [Die+12; KSP08]. Applying WBC to a mobile assistive robot reduces the need for the user to manually coordinate the wheelchair's movement with those of the robot arm. Thus, it facilitates the execution of tasks that require a wide range of motion, such as opening doors or drawers.

2.1.2 Human-to-robot interfaces

Figure 2.1: **Interfaces for assistive robots. Left**: A 6D Spacemouse from 3Dconnexion. **Center**: A sip-and-puff interface used by the pilot winning the CYBATHLON Challenges 2023. Source: CYBATHLON Challenges 2023. **Right**: A Brain-Computer Interface used to manipulate a robotic arm. Source: [Hoc+12].

In assistive robotics, different types of human-to-robot and robot-to-human interfaces exist.

The choice of interface depends on the user's preferences and capabilities, with variations in the amount of information they provide (number of DoFs, continuous vs. discrete signals) and the quality of the commands (ranging from a noisy electroencephalography (EEG) to a precisely controlled 2-DoF joystick). Assistive systems must account for both the interface and user abilities, as a single assistance method is unlikely to effectively accommodate this wide range of interfaces.

Power wheelchairs are commonly controlled with 2 DoFs commercial joysticks, which are, however, not adapted to all users. Hence, various interfaces adapted to disabilities have been investigated; see [ORR19] for a review. We mention a few here.

The first category regroups non-wearable sensors such as:

- Graphical user interface on touchscreens, such as a tablet or a smartphone, evaluated in Chung *et al.* [Chu+17].
- Sip-and-puff interface, used for example by Broad *et al.* to control a hierarchical finite-state machine [BA16].
- Voice and gesture interface, used for example by Jiang *et al.* [Jia+16] for feeding and dressing tasks.

A second category is non-intrusive wearable sensors, which monitor motions and biosignals from different body parts. Inertial Measurement Units (IMUs) have been investigated for individuals who retain partial or complete control of their neck and shoulders. For instance, Padmanabha *et al.* [Pad+24] demonstrated the use of blended shared control to allow a participant to teleoperate the Hello Robot Stretch RE2 robot for tasks such as blanket manipulation and face wiping with IMUs. Similarly, Jain *et al.* [Jai+15b] combined IMU data with shoulder motion tracking, enabling the user to control the manipulator speed and switch between a set of available trajectories. Choudari *et al.* [Cho+19] demonstrated wheelchair control based on electrooculography, where electrical potentials related to eye motion are measured. Kim *et al.* [Kim+13] tracked tongue motions with a magnetic tracker of eleven participants with spinal cord injury for computer and wheelchair control.

Another common interface is electromyography (EMG), the measurement of muscular activity, which is used in a wide range of application fields including myoelectrical prosthesis [Fle+21], rehabilitation [BKM16], computer gaming [Nac+11], and teleoperation for space applications [Hag+21]. EMG has also been used to create control interfaces for robotic manipulators for able-bodied users [VCS11; AK10; Iso+15].

Several studies have explored multimodal interaction by combining different interfaces. Fall *et al.* [Fal+18] combined IMU and EMG-based interfaces for pick-and-place tasks, while Baldi *et al.* [Bal+17] designed a cap integrating both sensor types and tested it on insertion tasks. Nam *et al.* [Nam+13] tracked tongue movements, eye movements, and teeth clenching to issue commands to a humanoid robot.

Brain-computer interfaces (BCIs), which extract commands from brain signals, have also been explored as an input method for assistive technologies. BCIs are either *non-invasive* – when the sensors are placed on the skin, as in [Ras+20] – or *invasive* – when the sensors penetrate the skin or are permanently implanted. Among non-invasive methods, EEG is the most widely used approach [OA13], with promising results demonstrated in robotic arm control [Men+16]. However, controlling systems with many DoFs remains challenging due to lengthy training times, user fatigue, and the low signal-to-noise ratio of EEG signals [Abi+19]. In contrast, invasive BCIs offer higher signal bandwidth and spatial resolution [Van+09], which is advantageous for extracting high-dimensional commands. They also enable more intuitive control schemes, as demonstrated by Hochberg *et al.* [Hoc+12], where neural signals related to motor imagery of a limb were used to control a robotic arm.

Robot-to-human interfaces include displays, sound, or haptic feedback. A display can provide helpful information to the user, such as world model information, the current controlled device and control mode, feedback on which commands are being given, etc. See subsection 3.3.2 for an example on EDAN. Using haptic feedback can be challenging for people with motor impairments when sharing control of a robotic arm due to reduced sensory perception and increased cognitive load.

2.2 Shared control

Shared control of assistive robots involves collaboration between a human user and a robotic system to achieve a task, with varying levels of control distributed between the two. This approach enhances the user's ability to interact with the environment while reducing cognitive and physical burden. Unlike other levels of autonomy (see section 1.3), shared control allows the user to provide continuous motion commands to the robot, with various control aspects coming into play. It is possible to assist the control of the wheelchair [BA16; KD18; Dev+19] or the manipulator.

For the latter, one approach is adaptive authority allocation from a control perspective, such as force-level control [Bal+20] or multi-level approaches [IDA21]. Another approach is when the assistance system generates commands combined with the user's inputs, a process formalized as "policy blending" [DS13]. These commands may be derived from a planner based on specific cost functions or through learning from demonstration. Another approach involves adjusting the mapping from user commands to end-effector movements. This can be optimized according to a cost function [HHS16] or to replicate demonstrated trajectories [Los+22]. It is also possible to enforce task-space restrictions [AM97; AEK05] by constraining the position or applying forces and velocities to the manipulator. In teleoperation settings such as surgical procedures, haptic feedback can also play an essential role [BDB13; Hag+24].

Shared control can also incorporate user intent modeling, predicting the task the user intends to perform. This can be used, for example, to adjust command blending [JA19; DS13; Jav+18] or impose task-space restrictions [MHD16; Got+22; IDA21]. The following sections provide a more detailed exploration of these different aspects.

2.3 User intent estimation and exploitation

An important aspect of realizing shared control is considering the user's goal or intent when using an assistive system. Typically, this intent revolves around interacting with specific objects, such as picking up a mug or opening a door. Once an object is picked, the user may choose to perform different actions, like drinking from the mug or placing it elsewhere. Various models have been proposed to estimate the user intent and exploit it in different ways.

One approach is to assist users in selecting actions when multiple options are available. The simplest approach is to use the distance between the end-effector and objects as a likelihood, then start the assistance once they are close enough. Naughton *et al.* [NH22] proposed structured task predictions for dynamic environments learned from demonstrations. Instead of directly predicting the desired task, each candidate task is evaluated on a common scale, and the most relevant ones are presented to the user via a Graphical User Interface (GUI).

Intent inference can also be employed to restrict the task space. Mehr et al. [MHD16] compute and enforce simple geometric constraints online during shared control execution of tasks. Iregui et al. [IDA21] restrict the task space and adjust the mapping of user commands. Muelling *et al.* [Mue+17] use intent inference to select a grasp pose among a set of stable, feasible grasps associated with the target object. Then, they constrain the end-effector with capture envelopes, a type of active constraint, to guide it to a grasp.

Intent inference can also be used to adjust the blending of commands between the user and a policy, as detailed in the following subsection.

2.3.1 Policy blending

Policy blending, formalized by Dragan *et al.* [DS13], refers to a framework where user commands are combined with assistive commands from an arbitrary policy, with a trade-off factor often influenced by

the estimation of user intent. Assistive commands can take various forms, such as following a trajectory based on a cost function [Jav+18], providing collision avoidance for dynamic targets using potential fields [Got+22], following an online trajectory generated through trajectory planning [Hau13], or adhering to a learned task model [Qia+21; Abi+17].

Jain *et al.* [JA19] explored user intent estimation and policy blending across multiple control interfaces, which vary in terms of the continuity and dimensionality of the control signals. Their work highlights the benefit of incorporating human-agent behavior as goal-directed actions with an adjustable rationality model tailored to individual users. Another notable approach is presented by Gopinath *et al.* [GA20], where the assistive system is optimized to actively disambiguate the user's intent, improving the overall effectiveness of the assistance.

Additionally, Dragan *et al.* [DS13] evaluated the effect of assistance aggressiveness, noting that strong assistance can be highly beneficial when correct but detrimental when incorrect. Their study demonstrated that factors such as the level of aggressiveness, the accuracy of predictions, and the complexity of the task significantly influence the assistance performance and user acceptance. They also introduced various metrics to evaluate the trade-off between user and assistive system inputs, such as the probability assigned to the system's predictions.

2.4 Input mappings

The end-effector of an assistive robot typically has 7 DoFs in task space, six for the end-effector pose and one for the gripper. In contrast, most input devices for motor-impaired users usually offer 2 or 3 control DoFs, as seen in subsection 2.1.2. Therefore, mapping these user commands to realize end-effector velocities and provide intuitive and efficient control is a crucial challenge. Since no standardized terminology exists, we will refer to this concept as "input mapping" and propose a formal framework for it in section 4.2.

2.4.1 Direct control

A common approach to controlling an assistive robot's 7 DoFs is to directly map each DoF of the user's input command to a velocity command applied to a specific DoF of the robot's end-effector. When controlling only a subset of the end-effector's DoFs, mode switching can change which subset is controlled. For instance, with a 3 DoFs control input, users can select different control modes: translational, rotational, or gripper control. Mode switching is often triggered by a signal, such as a button press. Rotational control can be referenced either to the robot's base frame or the end-effector frame, though both methods tend to be unintuitive for users. Work such as Campeau *et al.* [Cam+18] proposed a new dynamic frame (independent from the environment) for more intuitive control and demonstrated improved performance with 25 able-bodied users. Tijsma *et al.* [TLH05] investigated a new method to switch modes and different frames on which to apply rotational velocity, testing them with able-bodied and motor-impaired participants. Despite these advancements, direct control is often time-consuming, even for comparably simple tasks, and tasks that require synchronous motions – in translation and rotation – are challenging to execute, creating a high mental load.

If the robot can access a world model, task-specific automatic mode-switching becomes possible. For instance, Herlant *et al.* [HHS16] proposed automatically switching the controlled DoFs based on a time-optimal metric for tasks like dialing a rotary phone, pouring water, and unscrewing a jar, using a 2D joystick. In another example, Muelling *et al.* [Mue+17] implemented a system where translational control was the default, with task-specific switching to rotational velocity mapping for a *Pour* task, in an experiment with a BCI interface.

2.4.2 Learned input mapping

Learning from demonstration is a technique in which a robot learns to perform tasks by observing and imitating human demonstrations rather than being explicitly programmed. This method has also been explored to facilitate the design of input mappings. For example, Losey *et al.* [Los+22] utilized a conditional variational auto-encoder to approximate 6-DoF end-effector trajectories using latent actions applied on a trained decoder. They formalized the necessary properties of the latent space to reproduce such trajectories and combined this approach with policy blending and intent estimation. The system was evaluated across various tasks using 2D joysticks with both able-bodied and motor-impaired users.

They identified four properties for user-friendly latent actions:

- Conditioning: Latent actions should adapt to different contexts. For instance, the user needs control over different DoFs depending on whether they intend to open a drawer or pour a liquid.
- Controllability: The latent action space should enable the user to move the robot between arbitrary start and goal states within the demonstrated tasks. This measures how well the user can cover the range of demonstrated data.
- Consistency: The same latent action should produce similar effects on the robot's behavior in nearby states, ensuring predictable outcomes.
- Scalability: Smaller user inputs should result in smaller robot movements, while larger inputs should cause larger motions. Direct control naturally satisfies both consistency and scalability.

In addition, they aimed for orthogonal commands, enhancing the control interface's intuitiveness.

Przystupa *et al.* [Prz+23] argue that an input mapping should be locally linear and orthogonal, and that learning an input mapping explicitly under these conditions is more advantageous than learning it implicitly via losses of a latent space. To this end, they proposed 'action maps', which conditionally map user commands to joint velocities commands based on the robot's joint state. In their comparison of action maps, direct control, and latent actions on *pick* and *pour* tasks, they found that state-conditioned linear mappings performed better than latent actions. Surprisingly, they also noted that direct control was at least as effective and was preferred by some users. However, their approach shows drawbacks: the method offers joint control rather than task-space control, which limits reversibility and generalizability outside the training distribution.

2.5 Active constraints

Another way to assist the user is by limiting the accessible workspace of the manipulator or its end-effector, which can be achieved by 'active constraints'. These constraints are dynamically enforced and guide or restrict the system's motion in specific directions or within certain boundaries, making the task easier or safer to complete. For instance, they can prevent the manipulator from entering the user's immediate vicinity to ensure collision-free interactions or keep a grasped glass upright to avoid spilling water. Additionally, active constraints can support the user by guiding the end-effector towards a desired position, funneling all possible trajectories toward a pre-grasp pose, or ensuring the end-effector remains pointed at a target object (e. g. a bottle) when the user's intent to grasp it is detected.

2.5.1 Active constraints with virtual feedback

There is a substantial body of literature on the use of active constraints. In surgical robotics, users can either interact directly with the tool-carrying manipulator or teleoperate the system, where haptic feedback is often employed to enhance control. Active constraints are typically applied to either guide the user along a specific task pathway or restrict the user to a defined 'safe' region.

Previous research in the surgical domain, as reviewed by Bowyer *et al.* [BDB13], identifies three steps for implementing active constraints on a robot:

- Constraint definition, such as using surface or volumetric primitives, point clouds, or meshes.
- Constraint evaluation, where the system typically determines the closest point on the defined constraint.
- Constraint enforcement, which can be achieved through methods like proxy-based techniques with elastic linkages or potential fields.

These constraints can vary widely, depending on the intended application, such as being regional or guidance-oriented, attractive or repulsive, static or dynamic.

Selvaggio *et al.* [Sel+18] proposed an active constraints online generation technique based on the interaction force measurements. Rahal *et al.* [Rah+19] employed predefined constraints for robotic cutting and conducted a user study, finding that while performance improved with increased assistance, users reported disliking feeling more restricted and less in control. Balachandra et al. [Bal+20] implemented constraints directly at the force level, exploring how constraints can be tuned for better user experience.

2.5.2 Learning constraints

Some studies focus on learning constraints from demonstrations. Ahmadzadeh and Chernova [AC18] proposed to use generalized cylinders – surfaces with smoothly varying cross-sections – to generate autonomous behavior from trajectories, which can also be used to constrain the end-effector. Zeestraten *et al.* [ZHC18] first learned an assistive policy from demonstrated data. Then, they used policy blending between user commands and the assistive policy by computing a covariance matrix for both, using a Gaussian Mixture Model (GMM). An additional Gaussian with high variance is introduced to deactivate assistance when the system moves away from the demonstrated data. This method was tested on a realistic use case: unscrewing a cap at CERN. Similarly, Michel *et al.* [Mic+21] use learning from demonstrations with GMM to learn the impedance controller stiffness based on contact forces. In [Rai+18], Gaussian mixture models encode constraints from demonstrations using precision matrices as a proxy to control stiffness. The assistance guides the end-effector along multiple virtual guides, selected via the variance observed in the demonstrations.

Fitting a single parametric model to data can be done with an algorithm like RANSAC [FB81]. More advanced methods are required for multi-model fitting, a well-known problem in computer vision, as discussed by Delong *et al.* [Del+12]. Their proposed energy minimization method iteratively fits multiple geometric models (such as cones, cylinders, and lines) to the data. Data points are assigned to a model in each iteration or identified as outliers. Another approach for learning constraints, aimed at simplifying the recording process, is presented by Subramani *et al.* [SZG18]. They demonstrate that even if there is a static offset between the recording and constrained points, constraints such as lines, planes, axial rotations, or fixed joints can be learned.

2.5.3 Constraints for control

Another line of research using constraints for tasks specifications are the iTaSC – Iterative Task Specification and Control [Smi+08] – and Etasl – Expression-based Task Specification Language [AD14] – frameworks. iTaSC is a model-based framework that represents tasks as an optimization problem in terms of constraints on motion, forces, and control. Bartels *et al.* [BKB13] used iTaSC to solve the task of pancake flipping by defining geometric constraints using differentiable feature functions. eTaSL is a more generic, flexible, and expression-based framework for task specification. It extends iTaSC by providing a high-level language for specifying task goals, priorities, and the relationship between different task parameters.

2.5.4 Constraints for planning

In the planning domain, Berenson *et al.* [BSK11] proposed Task Space Regions (TSR), defined as a volume in SE(3) with tolerance in position and orientation (defined as roll, pitch, yaw) and a static transformation to a constrained frame. TSR was originally designed for a sampling-based planner and allows rejection, projection, and direct sampling. Various constraints can be represented for tasks such as grasping objects or manipulating articulated objects. Another work in this domain is by Chou *et al.* [CBO21], who focus on identifying a global constraint shared across tasks. By assuming bounded suboptimal demonstrations and having access to cost functions and dynamics, the authors explore both the forward problem (generating new demonstrations) and the inverse problem (recovering constraints). Aiming for full autonomy in operating articulated objects, Phillips *et al.* [Phi+16] use demonstrations represented as Experience Graph [Phi+12] to constrain and thereby expedite the planner's search.

2.5.5 Constraints for shared control

Finally, some research focuses on the use of constraints for shared control. Vogel *et al.* [Vog+16] address *pick* tasks by assisting with a range of possible grasps, guiding and orienting the end-effector toward the object, and automating the grasping process once triggered. Muelling *et al.* [Mue+17] employed constraints named 'capture envelopes' to guide a BCI-controlled end-effector toward grasping poses. Perez *et al.* [PS17] propose C-Learn, a constraints learning algorithm targeted at bimanual robots, where a multi-model representation is used to build a library of skills from demonstrated data, which can then be reused. Task Space Regions, posture and trajectory constraints are used. Keyframes, a sparse set of poses that accomplish a learned task if executed in sequence, are used to segment data. Shared autonomy is provided in the sense that users can review the motion plan and adjust keyframes offline if necessary. Works such as Mehr *et al.* [MHD16] focus on inferring constraints on the fly without making assumptions about the environment, allowing for real-time adaptability. Iregui *et al.* [IDA21] share a similar goal as ours for providing assistance and present a reconfigurable, adaptable, and modular assistance method based on Etasl. The framework considers multiple types of interfaces, user intent estimation, autonomy level modulation, and reactive control. A comparison between their method and our contribution is proposed in section 8.2.

2.6 Skill representation as finite-state machine

Our approach uses some of those constraint models mentioned above in a multi-model representation. This can be achieved by constraining the end-effector with multiple models at once, for example, on different DoFs, or by having different skill representations in multiple sections corresponding to different phases of the task. Our shared control skills are represented as a finite-state machine, a type of graph with a finite set of states connected by transitions, with one state active at a time.

Finite-state machines are widely used in robotics. Brunner *et al.* introduced Rafcon [Bru+16], a graphical interface designed for constructing hierarchical state machines. In the area of human-robot collaboration, Willibald *et al.* proposed a collaborative robot programming framework in [WEL20], which incrementally generates and refines a graph to structure a task's probabilistically encoded states. This idea was further developed in [WL22], where they incorporated intention recognition and feature clustering to infer individual feature constraints for each skill, enabling multimodal anomaly detection when generalizing the skills to new setups.

Manschitz *et al.* [Man+14] use a graph structure with merging properties for representing sequences of robot movements for a *lightbulb unscrewing* task. Transitions were learned with a Support Vector Machine. Niekum *et al.* [Nie+15] used Bayesian non-parametric statistics to detect repeated structures across multiple task demonstrations, identifying task invariants, key features, and the high-level task

structure. These skills were represented with a finite-state machine and executed as Dynamical Movement Primitives (DMP) on tasks such as *pick and place* and *letter drawing*.

Mohseni *et al.* [Moh+19] learned task structures as hierarchical task networks, while constraints were captured as TSRs during ongoing vocal interactions with the robot during demonstrations. Zhang *et al.* [ZZZ20] learned a cloth-folding task in simulation to have full access to the ground-truth data with its hierarchical structure, which is then executed on a real robot. This skill was represented using an 'And Or Graph', learned through grammar induction. Lastly, Kroemer *et al.* [KNK21] extensively reviewed the literature on learning for manipulation, arguing about the usefulness of a hierarchical representation for skills and presenting various data segmentation approaches.

Literature on using a graph or finite-state machines for shared control include, for example, the work of Shafti *et al.* [SOF19], where gaze estimation along with finite-state machines was used to assist *pick*, *place*, and *pour* tasks. Broad *et al.* [BA16] employed a sip-and-puff interface combined with a hierarchical finite-state machine to offer wheelchair control assistance. Park *et al.* [Par+20] proposed feeding assistance in shared autonomy, with three skills – scooping, feeding, and wiping – represented as a finite-state machine. Hagmann *et al.* [Hag+24] used shared control skills – represented as a finite-state machine – with haptic feedback and adaptation of the level of autonomy for two surgery training tasks: peg transfer and suturing.

2.7 Interactive imitation learning

Several studies on learning from demonstration, also known as imitation learning, have been discussed in previous chapters. These approaches aim to simplify the design of new robot behaviors through human instruction. These methods typically consist of two phases. The first is the learning phase, where demonstrations are recorded, key information is extracted, and assistive skills are developed. The second is the execution phase, where the learned skills are applied to assist with specific tasks. However, since user preferences and target environments, such as a user's home, may change over time, it is also beneficial to adapt the assistance behavior continually.

Interactive imitation learning is a branch of imitation learning in which human feedback is intermittently provided during the execution of a robot policy, enabling online improvement of the robot's behavior or in between executions, providing continual adaptation. This topic was reviewed by Celemin *et al.* [Cel+22], where different types of feedback, interfaces, and models were considered. Feedback can be either absolute (e. g. "this is the best trajectory") or relative (e. g. "this trajectory is better than the other"). It can be provided in either the evaluation space (the result of execution) or the policy space (how to best perform individual actions in specific states). Either the human or the robot may determine the timing of feedback. Negative reinforcement with counter examples can also be used. We focus on two use cases in particular: corrections provided through kinesthetic teaching and corrections provided directly by the target user.

2.7.1 Correction with kinesthetic teaching

One strategy for adaptation is to provide new demonstrations or via-points to a model using learning from demonstration, for example, generalized cylinders [AC18]. Via-points can also be used with Gaussian processes (GPs) [RW06]. However, not modeling aleatoric uncertainties makes GPs less attractive to model constraints. Other probabilistic motion primitive approaches [Par+13] permit via-point adaptation but require the definition and parametrization of basis functions.

Ewerton *et al.* [Ewe+16] explored learning ball-reaching trajectories using a 4-DoF elastically actuated arm. The ball's position is used as context, and the desired trajectories – represented as torque commands through Probabilistic Movement Primitives (ProMP) – are iteratively refined. Some studies focus on more

complex contexts. For instance, Jain *et al.* [Jai+13; Jai+15a] proposed a set of features related to the manipulation task, environment, and the user within the workspace. They developed score functions to capture user preferences and conducted user studies involving *pick*, *place*, and *pour* tasks and two robots. They demonstrated that robots could be trained within minutes with only a few incremental kinesthetic teaching feedbacks from non-expert users.

Related work by Bajcsy *et al.* [Baj+17] also examined user corrections, which modify the robot's current trajectory. In their framework, the robot optimizes a reward function that balances task completion while minimizing human effort. Task objective parameters are treated as hidden, with physical interactions serving as observations to infer these parameters. User corrections help adjust features such as the orientation of a cup, the distance to a table, or avoiding obstacles. An extension of this work [Baj+18] proposed that corrections should only affect a single feature to minimize unintended learning by the robot and ensure smoother task execution.

In assistive robotics, Canal *et al.* [CAT16] proposed a personalization framework for adaptive robotic feeding assistance. Their approach starts with a base skill, which is then customized for the user by a caregiver. Kinesthetic corrections refine a movement primitive modeled as ProMPs, enabling adjustments to both the feeding position and the distance relative to the user. In the haptic domain, Abi-Farraj *et al.* [Abi+17] developed an assistance based on trajectories demonstrations, using Locally Weighted Regression. The assistance, represented as a virtual spring that generates forces with varying stiffness, adapts based on the deviation from demonstrated trajectories. Their model is further refined after each task execution, adapting to user preferences and thereby reducing the operator's effort over time.

If the system exhibits sensing redundancy, a whole-body contact estimation can be provided, which enables a sense of touch to the entire robotic structure [IAD24].

2.7.2 User corrections via a human-robot interface

In our use case of assistive robots, adaptive behavior by the users of shared control skills is achievable only through user interfaces, as kinesthetic teaching is impossible for the target user group. Mehr *et al.* [MHD16] developed an approach to infer end-effector constraints in real-time while the user performs a task. This method includes a confidence metric that evaluates how well the current constraint fits the task. It allows the system to adjust when users change their desired goal mid-assistance. Broad *et al.* [Bro+17] explored using natural language corrections to adjust planned motions. Selvaggio *et al.* [Sel+18] introduced a system where users in a surgical setting can dynamically generate and switch between active constraints with stiffness adaptation.

Kernelized Movement Primitives (KMP) models can be adapted with via-points, or by external signals [SH23]. We make use of this property in Chapter 7.

2.8 User experiments with assistive robots

2.8.1 Studies with motor impaired users

There is a substantial body of research involving able-bodied users and motor-impaired users in the context of assistive robotics. Studies with able-bodied users are usually easier to set up and can provide useful insights; however, they are not as directly relevant as studies with the target population. Motor-impaired users have specific needs and interact with assistive systems differently than able-bodied users. For instance, they may rely on alternative input modalities (e. g. EMG, sip-and-puff or voice) that able-bodied users might not find necessary. They might also face difficulties such as prolonged task execution time, increased cognitive load, fatigue, or discomfort during extended use. Hence, studies with motor-impaired users are essential for validating assistive technologies' actual effectiveness and applicability.

For instance, Maheu *et al.* [Mah+11] assessed 34 motor-impaired users operating a Kinova JACO robotic arm using direct control. The tasks included picking and placing objects like bottles, tissues, and straws, pressing buttons, and pouring. The results demonstrated that, after a brief period of interaction with the robot and adequate training on its use, all participants (excluding three with interface problems) could complete all the tasks.

A set of works explores assistive feeding technologies. Bhattacharjee *et al.* [Bha+20] conducted a study on user preferences involving ten motor-impaired participants using either a GUI or voice control. The study examined the use of supervised autonomy during both individual and social dining scenarios. The assistance was divided into four phases: bite acquisition, bite transfer, bite timing, and bite delivery, with different strategies available for each phase. Participants tested three levels of supervised autonomy: high (with no user intervention), low (with the user selecting the strategy at each phase), and medium (with strategy selection in only two phases). The findings revealed that user preferences varied based on the context and the severity of impairment. For example, users favored faster assistance and voice commands in individual dining but preferred slower assistance and GUI-based control in social dining. The study also received positive feedback about the system's ability to provide greater independence. One user remarked, "There is a remarkable amount of independence with this (...) I like to be able to (eat) by myself". Another expressed interest in more autonomy, stating, "I didn't like having it do step by step. I wanna say give me food and have it fly down there and give me food".

Park *et al.* [Par+20] introduced a system that autonomously performs visually-guided actions to scoop or stab food and deliver it into a user's mouth, tested with nine individuals with motor impairments. Users can choose with a GUI between tasks such as scooping, feeding, or cleaning the spoon as well as adjusting the food delivery position. The robot can switch utensils depending on the type of food. The system was evaluated using metrics including task completion time, ease of use, comfort, reliability, and safety. Their findings highlight the importance of tailoring the system to user preferences, as participants demonstrated varying preferences regarding aspects such as interface design and operation speed.

Muelling *et al.* [Mue+17] evaluates the performance of shared control in combination with a BCI in the rehabilitation benchmark tasks of the ARAT, box-and-blocks test, door opening, and pouring. Their system integrated multiple aspects for assistance: a world model, used by the user intent estimation with maximum entropy inverse optimal control [Jav+18], blended commands with a prediction confidence of the user's goal, input mapping for the *pour* task, capture envelopes as constraints and a compliant controller for safety. They tried to keep the human operator in control to the largest extent possible and allow the user to break away from the assistance.

A motor-impaired user, Henry Evans, participated in the 'Robots for Humanity' project, which spanned several studies [Cio+12; Che+13; GK19]. This project emphasized the value of involving care recipients and caregivers in the participatory design process. Using a head-tracker cursor as input to a GUI and varying degrees of autonomy, Evans was able to perform a sequence of tasks with a PR2 robot, such as grasping a cabinet handle, opening a cabinet door and drawer, retrieving a towel, and navigating to a designated drop-off point. Assistance was also provided for tasks involving manipulation close to the user's body (scratching and shaving), giving candy to children on Halloween, and collision-free navigation. The researchers concluded that effective training mechanisms, appropriate interfaces, and varying levels of autonomy are crucial for enabling assistive robots to support with daily living tasks in real homes.

Javaremi *et al.* [JA20] conducted a study using a Kinova Jaco arm mounted on a table to investigate how to automatically adjust the autonomy level of assistive robots. The study involved eight motor-impaired and twelve able-bodied participants and focused on developing metrics based on task difficulty and the type of interface used. The tasks included placing a butter knife inside a drawer, scooping cereal, picking and placing a plate, mug, and cereal box, unscrewing a jar, and pouring cereal.

2.8.2 Studies with able-bodied users

Although motor-impaired user studies are more directly relevant, able-bodied studies still play a valuable complementary role in methods and systems development. Belkhale *et al.* [Bel+22] evaluated an assistive feeding system with six able-bodied participants. They proposed a balance between comfort and efficiency, formalized as heuristics and integrated into motion planning. Their results demonstrated that incorporating these heuristics significantly impacted the generated trajectories, providing significantly more preferable trajectories compared to a fixed pose baseline. In Canal *et al.* [CTA21], 30 healthy participants performed three tasks: assisted feeding, shoe-fitting, and jacket dressing. In 70% of cases, users could identify whether their preferences were considered. Incorporating user preferences led to improved robot behavior and enhanced the overall user experience.

2.9 Simulation

Several simulations for caregiving robots have recently been developed. In addition to typical simulation challenges, these must address the complexity of having a user in the loop, particularly modeling user behavior and preferences.

Ye *et al.* [Ye+22] introduced RCare World, which incorporates high-fidelity user models based on clinical data, using behavior trees to simulate users' physiological conditions and behaviors. They designed a set of tasks, informed by professional occupational therapists, to ensure meaningful assistance: feeding, bathing, dressing, limb repositioning, opening doors, and lifting toilet lids. The simulation uses a Kinova Gen3 robot mounted on a wheelchair and provides planning and reinforcement learning (RL) algorithms. The researchers demonstrated successful skill transfer from simulation to real-world applications with a bed-bathing task. Assistive Gym by Erickson *et al.* [Eri+20] is another physics-based simulation framework for assistive robotics focusing on RL approaches. Its tasks include itch scratching, bed bathing, drinking water, feeding, dressing, and arm manipulation. Shervedani [She+23] proposed a simulator focused on human-robot interaction, particularly interacting with the elderly in a home environment, which they use to train an RL agent. Pascher *et al.* [Pas+24] used Unreal Engine to simulate a Kinova arm, incorporating visual cues such as arrows to indicate what the robot intends to do or can do. While they provide assistance through an adaptive input mapping method, the system does not simulate grasping physics or robot dynamics.

2.10 Conclusion

From this review, a few design choices were made. The control interface selection was beyond the scope of this thesis, and we present results using the provided 3D continuous interfaces: a Spacemouse and an EMG-based interface. Since these interfaces offer a higher number of DoFs than most alternatives, blended control methods were not required. Instead, on one hand we apply input mappings to make optimal use of the interfaces. On the other hand, the assistance system does not issue commands but rather constrains the pose of the end-effector, enabling users to maintain good control over the robot's movements and enhancing their sense of agency. Many everyday tasks have an inherent structure with distinct phases, such as approaching a drawer handle, grasping it, and pulling the drawer open. These phase transitions can be accurately modeled, which led to our decision to use a finite state machine for skill representation. Manually designing task constraints can be time-consuming, so we explored two approaches to learning from demonstration. One of these methods also supports constraint adaptation for personalization. Lastly, to ensure comprehensive evaluation, the SCT framework was tested with both able-bodied and motor-impaired users in various environments, including our laboratory, participants' homes, and CYBATHLON events.

Chapter 3

Background: EDAN

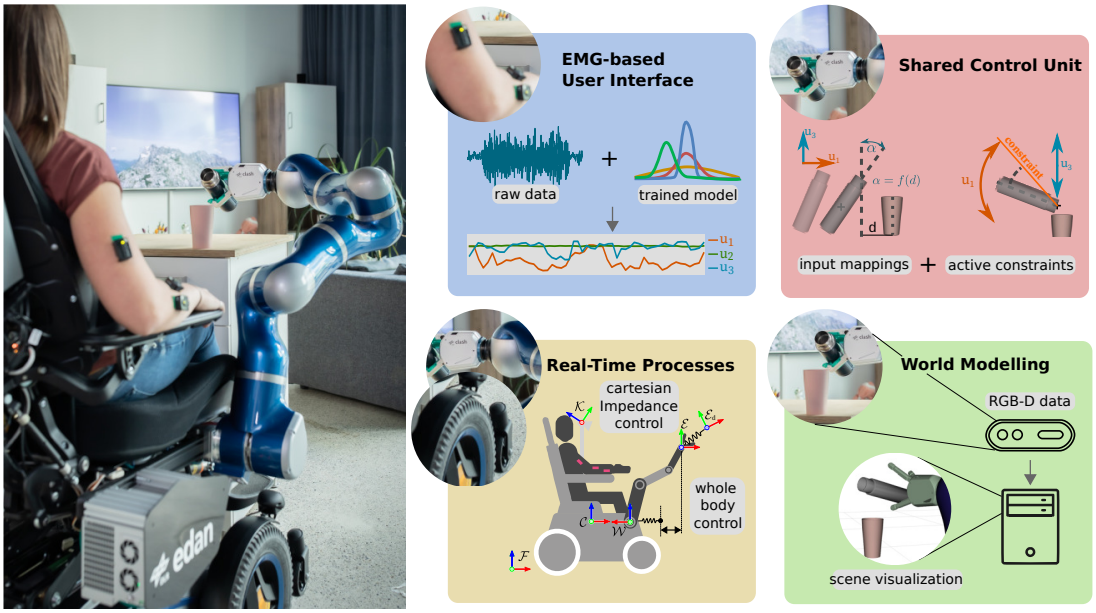


Figure 3.1: **The EDAN system.** An overview of the system and its four main components. The **EMG-based user interface** enables people with severe amyotrophia to perform 3D robot control, a **shared control** scheme based on SCT supports the user during complex tasks with the information from a **world model**, and **whole-body cartesian impedance control** takes care of the coordination of wheelchair and manipulator motion. Adapted from [Hag+25]

EDAN, the 'EMG-controlled Daily AssistaNt,' serves as a research platform for re-enabling robotics in areas such as human-robot control interfaces [VH18; HV18], whole-body control [Isk+19], and assistive control [Vog+16; Que+20]. This robot is the product of collaborative efforts within the Re-enabling Robotics group at DLR-RMC, and its various components are illustrated in Fig. 3.1. These components provide context for my contribution, which focuses on assistance through shared control, explored in detail in the next chapters.

3.1 Hardware

EDAN consists of a commercially available wheelchair on which a modified DLR Light-Weight Robot III is mounted, of which an overview is presented in Fig. 3.1. The manipulator was modified by adding an

8th joint at its base. This additional revolute joint increases the reachable workspace, especially in front of the user's leg space, so objects can be picked up from the ground and drinks or food brought to the user's mouth. The manipulator is equipped, for grasping and manipulating, with either a dexterous torque controlled five-fingered DLR-HIT hand or a three-fingered DLR-CLASH hand with impedance control and intrinsic compliance capabilities [FR20]. The wheelchair is equipped with additional magnetic-type encoders to measure the position of the actuated wheels, providing odometry for the mobile platform. It is controlled via the proprietary R-NET interface, through which forward-backward and rotational velocity commands can be sent with an analog signal.

All the computing of the robot software modules is done onboard. A Linux-based real-time computer (Intel I7 4-Cores) runs the low-level control software. The high-level control software uses another Linux-based computer (Intel I7 8-Cores). Object detection and localization are performed on an Nvidia Jetson TX2 embedded GPU in combination with an Azure Kinect RGB-D Camera. Processes for the DLR-CLASH hand are processed on an additional ATOM-based embedded computer.

An overview of the individual software modules is illustrated in Fig. 3.2 and explained below.

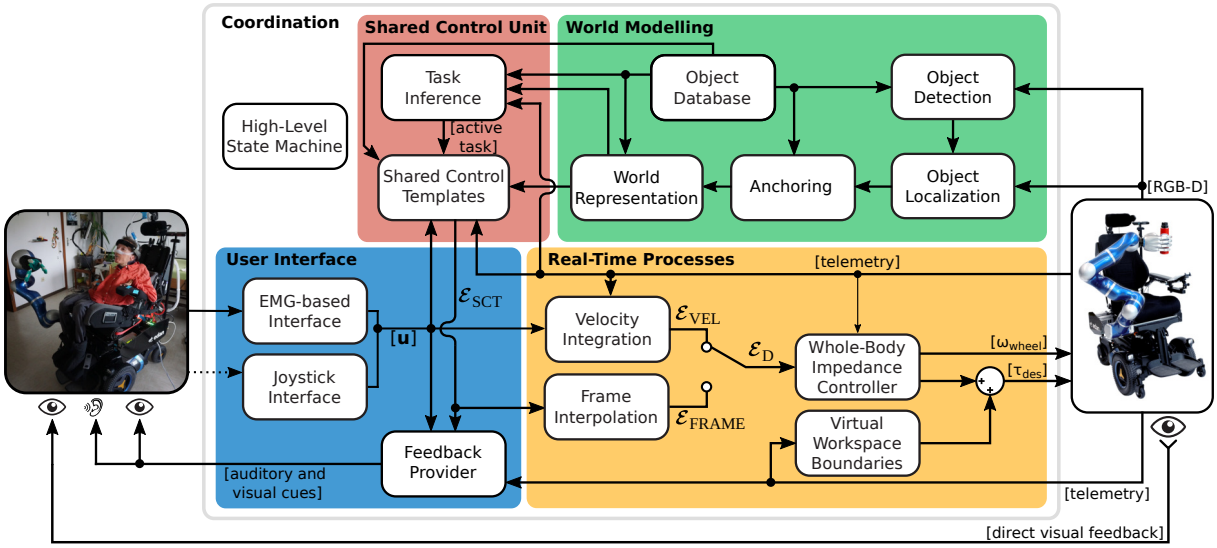


Figure 3.2: **System diagram, with software infrastructure and control modes.** In direct control, the desired end-effector position and orientation for the whole-body controller ($\mathcal{E}_{\mathrm{D}}$) is determined through velocity integration ($\mathcal{E}_{\mathrm{VEL}}$). In shared control, it is determined through frame interpolation ($\mathcal{E}_{\mathrm{FRAME}}$) based on the desired end-effector pose computed by the Shared Control Templates ($\mathcal{E}_{\mathrm{SCT}}$). Adapted from [Hag+25].

3.2 World modeling

The system's world model includes a set of known objects detected in the world and geometric information required to manipulate them. The world model is linked to an object database, which includes relevant information for object detection, localization, and shared control tasks. EDAN can dynamically create a world model from a scene using an RGB-D camera. The objects are instantiated in the world model based on a two-stage perception pipeline: object detection for class categorization followed by object localization for precise 6D pose estimation. After that, an Anchoring algorithm assigns symbolic tags to specific instances. Finally, the world model is aggregated in a world state representation, which the different modules of the system can query. This process is illustrated in Fig. 3.3.

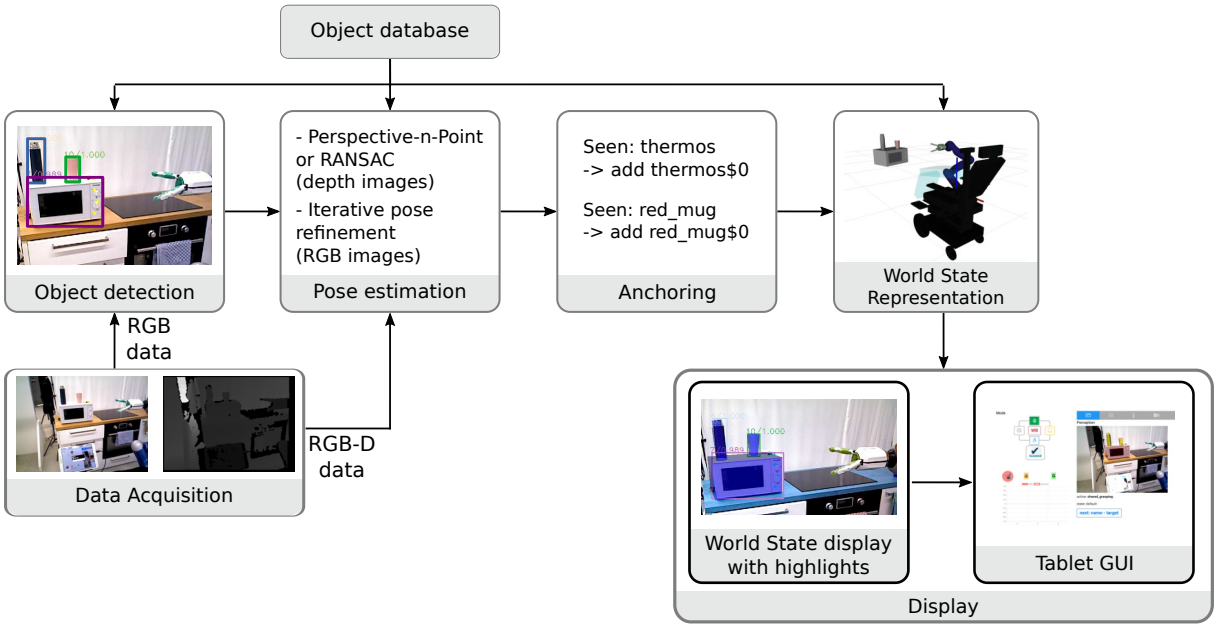


Figure 3.3: **Components of EDAN's world modeling module.**

3.2.1 Object database

The Object DataBase (ODB), first introduced by Leidner *et al.* [LBH12], stores all available object information accessible to the robot *a priori*. This includes attributes (e. g. color, mass, length), reference frames (e. g. the tip of a bottle relative to its origin frame), robot-related properties (e. g. grasp configurations), and 3D models. A key feature of the ODB is its object-oriented paradigm: physical objects (such as EDAN's red_mug) are derived from abstract classes (such as _mug, which itself inherits from _container). This enables *polymorphism*, which is leveraged to store shared control skills as *YAML* files. For example, a generic skill – such as releasing a cylindrical container – can be inherited by different objects (e. g. red_mug and thermos) while being instantiated with specific properties and reference frames.

3.2.2 Object detection and localization

The detection and subsequent pose estimation of target objects are achieved through the following steps:

- Object detection: First, regions of interest are identified using the YOLOv7 convolutional neural network, which offers real-time and highly accurate performance [WBL23]. The YOLOv7 model is trained on a combination of artificially generated data using the BlenderProc library [Den+23] and real images captured in various use case scenarios and environments.

- Pose estimation: The identified regions of interest, along with their respective images, are then processed using the algorithm proposed by Ulmer *et al.* [Ulm+23], which returns a 6D pose estimation for each detected object. For specific objects that lie on planar surfaces, a plane is first estimated with the RANSAC algorithm [FB81] on the image depth information. The object's pose is then estimated by combining the region of interest with the plane.

- Pose refinement: Finally, the initial 6D pose is refined based on a tracking and refinement algorithm developed by Stoiber *et al.* [Sto+23].

3.2.3 Anchoring

Coradeschi and Saffiotti define anchoring as "*the problem of connecting, inside an artificial system, symbols and sensor data that refer to the same physical objects in the external world*" [CS03]. For a mobile agent like EDAN, objects with associated symbolic information must maintain consistency and persistence after the robot performs actions. The anchoring process ensures this by determining whether an object observed after a robot action or from a new viewpoint (after moving the wheelchair) is the same as previously detected or a different instance. Similarly, an object temporarily hidden by the manipulator is still the same. Keeping track of symbolic information is essential to provide useful assistance: if a mug was used to drink then released on a table, it will hold the 'used' and 'empty' flags. This indicates to the user intent estimation process (subsection 3.5.3) that the likelihood of the user wanting to immediately pick up the mug again is low. Without anchoring, the mug would be detected as a new object and immediately ready to be picked. With the end-effector close to the mug after just releasing it, the user intent estimation process would immediately activate the skill, confusing the user. We say of an object that it is *anchored* if it has been seen enough times within a time window, with consistent localization. When an object is anchored, it is assigned a unique ID such as *red_mug$0* or *drawer_handle$1*.

The anchoring process holds an internal state that tracks all the object detections as a list of seen objects, some of which are anchored. At each new detection, the anchoring checks the closest seen object of the same class, using Euclidean distance. If this seen object is closer than 5cm to the new detection, its pose is updated with the latest detection. Otherwise, a new seen object is created from this detection. A seen object instance holds a variance estimate σ of the quality of the detections. If the Euclidean distance from a seen object instance to the new pose estimate is higher than 2σ, it is considered an outlier, discarded, and the variance increases. If not, the new pose estimate is considered valid, added to the seen object buffer holding pose estimates, and the variance is recomputed. The new object pose is a sum of the last ten detections, eventually biased towards the more recent pose estimates. If used, weights are empirically selected based on the rate and stability of new pose estimates. This simple filtering method provides stable object pose estimates, even with imperfect localization.

This process may create two objects of different classes anchored next to each other or even at the same spot if two object classes are wrongly detected for the same object. Hence, all anchored objects are checked pair by pair at every loop to see if arbitration is required, i. e. if they are too close to one another. In this case, for each object the quality of the pose as well as the number of detections in the last 5 seconds are taken into consideration. Then, the least likely object is deanchored. Objects not perceived for the last 8 seconds are also deanchored. Finally, when a request to update the world model is received, the anchoring process updates the world state representation according to its current list of anchored objects. A deanchored object keeps its ID: if it gets deanchored due to occlusion by the manipulator but is then detected at the same spot, it gets reanchored with the same ID.

3.2.4 World state representation

The World State Representation (WSR), also introduced by Leidner *et al.* [LBH12], contains the aggregated world model of the robot. As seen in Fig. 3.3, it contains a list of all anchored objects, their *a priori* known properties such as size and associated skills, the robot's belief in their current poses, and the symbolic properties of the objects – expressed as Planning Domain Definition Language (PDDL) predicates [Gha+98]. For example, after a grasping action, the *red_mug$0* would obtain a predicate *(grasped red_mug$0 edan_arm)*.

The WSR is updated by the anchoring process upon request, which can be triggered by the user or during transitions, such as when shifting from controlling the wheelchair to controlling the manipulator. This update process provides a static representation, ensuring stable behavior throughout the rest of the pipeline. The WSR is the central information source for all other components within the shared control

architecture, including the user intent estimation, task inference and shared control processes, see Fig. 3.2.

3.3 User interfaces

Numerous interfaces are available for assistive robots, as outlined in subsection 2.1.2. These interfaces depend on the users' abilities, which may change over time due to factors like experience or medical conditions. Ideally, an assistive device would offer a variety of users – with different preferences and abilities – interfaces with corresponding support. EDAN provides multiple interfaces. First, a head switch allows users to toggle between controllable devices such as the robotic arm, the wheelchair, and the tablet. Second, users can send 3 DoFs continuous inputs and a binary trigger with a joystick or an EMG-based interface. Third, a graphical interface is displayed on a tablet mounted on the wheelchair.

A joystick is a standard interface as it is already used for controlling wheelchairs, for users with sufficient finger motor ability. Here, we provide a 3D joystick using a SpaceMouse from 3Dconnexion. However, as discussed in subsection 2.1.2, alternatives may be necessary, with an EMG-based interface being one such option.

3.3.1 EMG-based interface

An EMG-based interface is provided on EDAN, see [VH18; HV18]. It interprets muscular activity measured with surface electromyography to generate a continuous 3D control signal with Gaussian process regression. Additionally, a binary trigger signal is decoded through a Linear Discriminant Analysis classifier. This trigger signal can be used in various ways, such as switching between modes in direct control, starting a *release grasped object on surface* task with shared control, or automatically completing a grasp in shared autonomy.

3.3.2 Graphical user interface

A tablet mounted on the wheelchair provides the user with a GUI displaying all relevant information. Fig. 3.4 illustrates the GUI and its various components (A–F).

By operating the head switch, the user can cycle through different device activations: robotic manipulator, wheelchair or tablet. When the tablet is selected, its interface changes (Fig. 3.5), allowing the user to choose between different control schemes, such as shared or direct control. Additionally, a list of all possible tasks – generated based on the current world model – is presented. The user can select and initiate any task from the list, even if it is not the most likely task according to the user intent estimation. The EMG-based interface for controlling the robot works similarly when controlling the tablet: the user navigates by highlighting items with the $\boldsymbol{x}$- and $\boldsymbol{y}$-components of the control command, while the binary trigger acts as a click to select the highlighted item, such as control schemes or tasks.

3.4 Real-time processes

Controlling the end-effector uses two primary strategies, as shown in Fig. 3.2. In direct control, velocity integration is employed, where the user input is converted to a velocity command applied directly to the end-effector, computing $\mathcal{E}_{\mathrm{VEL}}$. In shared control, a desired pose $\mathcal{E}_{\mathrm{SCT}}$ is first calculated at a rate of 30 Hz. A recommendable frame interpolator then computes a smooth trajectory towards this dynamic target $\mathcal{E}_{\mathrm{SCT}}$ at 1 kHz, complying with predefined velocity limits for translation and rotation. In both cases, these commands are transmitted to a whole-body controller, which computes coordinated commands that integrate arm and wheelchair movements. To ensure user safety, a virtual environment is integrated into the control scheme, limiting the manipulator's workspace to prevent direct contact between the user

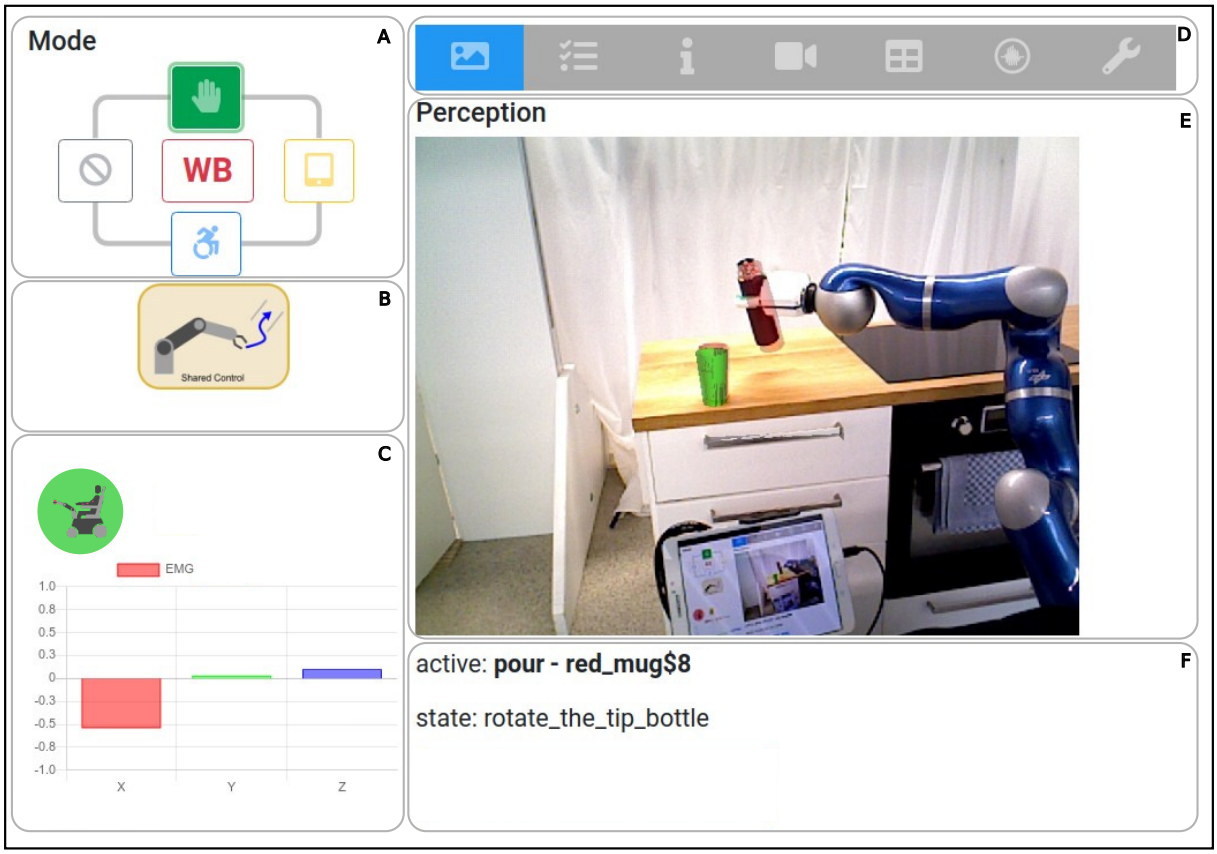


Figure 3.4: **GUI always available to the user.** Shown is the tablet view. **A:** Device controlled by the user, cycled through by operating the head-switch: 'robot control', 'tablet', 'wheelchair control', or 'nothing' (user commands have no effect to the system). Furthermore, [WB] is highlighted if the whole-body control scheme is active. **B:** Active control scheme (direct or shared). **C:** Decoded commands. The green circle provides information if the activity threshold is exceeded to allow a control input. **D:** Additional tabs display the list of control schemes, the list of tasks, and expert information not needed by the participants. **E:** World model visualization; shown is the RGB-camera stream augmented with the localized objects instantiated in the world model. Different colors highlight different states of the objects, like green, which highlights the target object of the current task. **F:** Information regarding the current task and states; shown is the active task as well as the current state of the task.

and the manipulator. This virtual environment generates repulsive forces, which are converted into joint torques and added to the control scheme.

3.4.1 Whole-body cartesian impedance control

Assistive robotic systems used in daily support scenarios frequently engage in continuous physical interaction with their environment. As a result, interaction forces between the robotic system and its surroundings are expected. Additionally, such a system interacts with humans, primarily its user, for tasks such as eating and drinking, as well as other humans, e. g. for a hand-over. Accordingly, a compliant behavior at the robot's end-effector is essential, i. e. the capability to adjust its motion and interaction based on external forces or environmental conditions, which can be achieved through active control [Isk+23]. The EDAN system features a reactive whole-body control that coordinates the movements of the wheelchair and the robotic arm, as detailed in [Isk+19]. This coordination enables the execution of

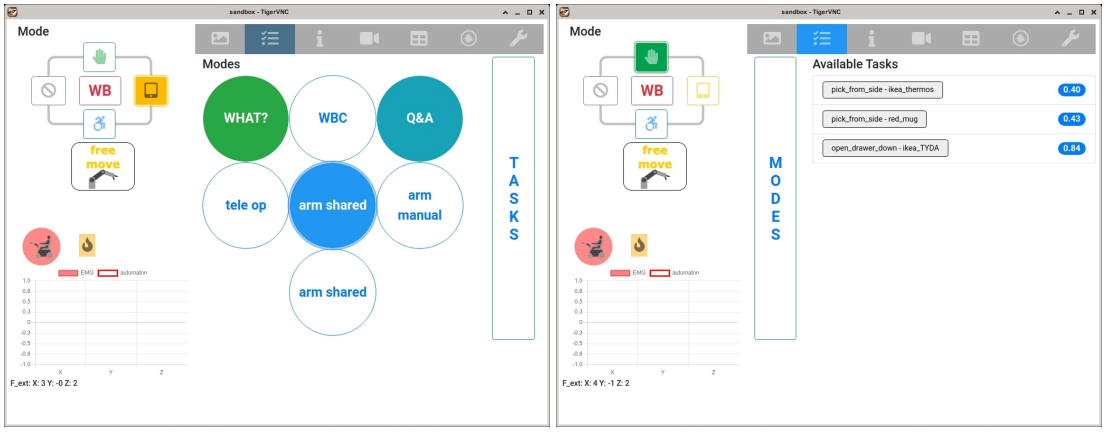


Figure 3.5: **Tablet GUI, when controlled by the user**. **Left**: Control mode selection. **Right**: Task selection.

tasks that extend beyond the manipulator's kinematic reach, such as opening a door.

To ensure passive physical interaction with the environment, we command the joint torques to realize Cartesian impedance behavior as

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{imp}} + \boldsymbol{\tau}_{\text{null}} + \boldsymbol{\tau}_g(\boldsymbol{q}), \tag{3.1}$$

where $\boldsymbol{\tau}_{\text{imp}}$ is the generated torque to actively realize the Cartesian impedance at the end-effector, the term $\boldsymbol{\tau}_{\text{null}}$ achieves a null space secondary task, and $\boldsymbol{\tau}_g(\boldsymbol{q})$ stands for the gravity compensation component with $\boldsymbol{q}$ the joints values. The Cartesian impedance control action can be formulated as

$$\boldsymbol{\tau}_{\text{imp}} = -\boldsymbol{J}^T(\boldsymbol{q})(\boldsymbol{K}_x\tilde{\boldsymbol{x}}(\boldsymbol{q}) + \boldsymbol{D}_x\dot{\boldsymbol{x}}) \tag{3.2}$$

$$\tilde{\boldsymbol{x}}(\boldsymbol{q}) = \boldsymbol{f}(\boldsymbol{q}) - \boldsymbol{x}_{\text{d}} \tag{3.3}$$

$$\dot{\boldsymbol{x}}(\boldsymbol{q}) = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{3.4}$$

The mapping $\boldsymbol{f}(\boldsymbol{q}) : \mathbb{R}^n \rightarrow \mathbb{R}^6$ encodes the forward kinematics with $n = 8$ on EDAN, while $\boldsymbol{x}_{\text{d}} \in \mathbb{R}^6$ denotes the desired task-space position and orientation. The Cartesian stiffness and damping are represented by the positive definite matrices $\boldsymbol{K}_x, \boldsymbol{D}_x \in \mathbb{R}^{6\times6}$, respectively. The controller relies on the low-level torque control loops to achieve the desired impedance. However, it is assumed that the torque dynamics are significantly faster, ensuring the intended impedance behavior is realized effectively [AOH07; Isk+20; Isk+22]. This whole-body Cartesian impedance control is illustrated in Fig. 3.6.

With this framework of hierarchical whole-body control, the elbow position can be regulated within the null space. The control action $\boldsymbol{\tau}_{\text{null}}$ is realized in the null space of the main end-effector task as

$$\boldsymbol{\tau}_{\text{null}} = -\boldsymbol{N}\boldsymbol{J}_{\text{e}}(\boldsymbol{q})^T\boldsymbol{f}_{\text{e}}. \tag{3.5}$$

Here, the task-space null space projector is denoted by $\boldsymbol{N} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{J}_{\text{e}}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$ and $\boldsymbol{f}_{\text{e}} \in \mathbb{R}^6$ are the elbow Jacobian matrix and control wrench, respectively. Similar to the primary end-effector task mentioned previously, the robot elbow control action is calculated to realize a desired Cartesian stiffness and damping such that $\boldsymbol{f}_{\text{e}} = (\boldsymbol{K}_e\tilde{\boldsymbol{x}}_e + \boldsymbol{D}_e\dot{\boldsymbol{x}}_e)$ at the elbow location $\boldsymbol{x}_e$. This control action can vary depending on the task or the environment [DO19; Die+21], see subsection 4.4.1.

3.4.2 Virtual workspace boundaries

Due to the functional requirements, the user has to be located within the robot's workspace to perform tasks such as bringing a filled drinking container to their mouth. To ensure safety, a virtual environment

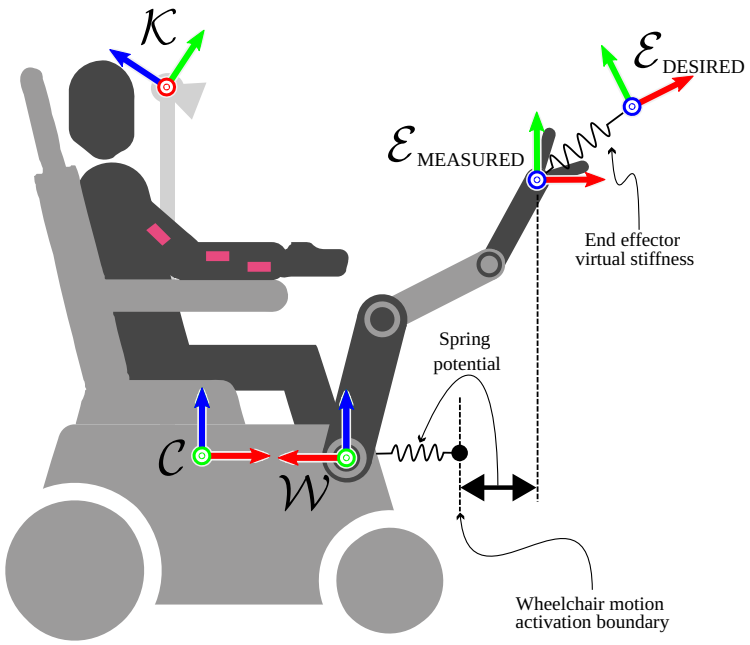


Figure 3.6: **Schematic illustration of the whole-body Cartesian impedance control.** The coordinate frames used by the system are highlighted. $\mathcal{K}$ is the camera frame, $\mathcal{C}$ the center of the wheelchair, $\mathcal{W}$ the base of the robotic manipulator, $\mathcal{E}_{\text{MEASURED}}$ the measured end-effector frame, $\mathcal{E}_{\text{DESIRED}}$ the desired end-effector frame. $\mathcal{W}t$ is the current pose of $\mathcal{W}$, while $\mathcal{W}0$ is a static frame initialized to $\mathcal{W}$ at the beginning of a skill. Adapted from [Hag+25].

is created to constrain the manipulator's workspace, preventing direct contact between the user and the manipulator.

This virtual environment generates virtual obstacle forces, which are transformed into joint torques and integrated into the control scheme. We therefore extend Equation 3.1 with $\boldsymbol{\tau}_{\text{d,VE}}$, which is the desired joint torque resulting from the virtual environment:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{imp}} + \boldsymbol{\tau}_{\text{null}} + \boldsymbol{\tau}_{\text{d,VE}} + \boldsymbol{\tau}_g(\boldsymbol{q}).$$

The virtual environment is composed of planes, spheres, and cylinders that define restricted zones for the manipulator, which are customized for each user to ensure safety. If the manipulator exceeds the workspace boundaries, the Cartesian end-effector velocity $\dot{\boldsymbol{x}}$, the current end-effector pose $\boldsymbol{x}$ and the the projected pose on the virtual environment's surface $\boldsymbol{x}_{\text{d,VE}}$ can be used to calculate a workspace boundary wrench $\boldsymbol{f}_{\text{V}i}$ as

$$\boldsymbol{f}_{\text{V}i} = K_{\text{V}i}(\boldsymbol{x} - \boldsymbol{x}_{\text{d,VE}}) + D_{\text{V}i}\dot{\boldsymbol{x}}_i.$$

Given N workspace boundary wrenches, the resulting torque $\boldsymbol{\tau}_{\text{d,VE}}$ generated by the virtual environment for the workspace limits is defined as

$$\boldsymbol{\tau}_{\text{d,VE}} = \sum_{i=1}^{N} \boldsymbol{J}^T \boldsymbol{f}_{\text{V}i}.$$

Additional workspace boundaries for other parts of the robot are defined using forward kinematics and the corresponding Jacobian matrix of the respective joint of the manipulator.

3.4.3 Safety trough compliant control

EDAN has a built-in safety mechanism, which reacts to unexpected external forces higher than a set threshold and puts the system in a maximally compliant control mode, in which only gravity compensation and virtual workspace boundaries are active. In this mode, the robotic arm is assumed to be highly backdrivable, which is ensured by observer or model-based methods that are employed at the joint level [WI18; IW19; Sch+24]. As a result, the arm cannot exert force or torque on the environment, providing safety to the users and the hardware. If the arm is not in contact with the environment, users can trigger a planned reconfiguration to restore it to its initial joint configuration.

3.4.4 Velocity integration

When using the direct control scheme to operate the manipulator, the user input $\boldsymbol{u}$ is interpreted as a velocity command applied to the end-effector. Depending on the subset of DoFs being controlled, this velocity command is integrated into either the translational or rotational component of the end-effector pose. For grasp selection, however, the user input is treated as a joint velocity command applied directly to the robotic hand's joint configuration. The user input $\boldsymbol{u} \in \mathbb{R}^3$, $\boldsymbol{u} \in [-1, 1]^3$ is a unit-less vector which is multiplied with the defined maximum control velocities k_{trans} in m/s, k_{rot} in rad/s or k_{grasp} in rad/s for translational, rotational or grasping motion, respectively.

$$\dot{\boldsymbol{x}}_{\mathrm{d}} = k\boldsymbol{u}, \tag{3.6}$$

$$\text{with } k \in [k_{\mathrm{trans}}, k_{\mathrm{rot}}, k_{\mathrm{grasp}}] \tag{3.7}$$

To also account for the workspace limits at the desired position and velocity level, we define the desired position $\boldsymbol{x}_{\mathrm{d}}$ as

$$\boldsymbol{x}_{\mathrm{d}} = \int_0^t [\dot{\boldsymbol{x}}_{\mathrm{d}} - \boldsymbol{c}(\boldsymbol{f}_{\mathrm{V}}, \dot{\boldsymbol{x}}_{\mathrm{d}})].$$

The state-dependent vector $\boldsymbol{c}(\boldsymbol{f}_{\mathrm{V}}, \dot{\boldsymbol{x}}_{\mathrm{d}})$ accounts for hitting workspace limits of the virtual environment by stopping translational integration of the decoded velocity signal if such a boundary is hit and the velocity direction points towards the wall:

$$\boldsymbol{c}(\boldsymbol{f}_{\mathrm{V}}, \dot{\boldsymbol{x}}_{\mathrm{d}}) = \begin{cases} \frac{-\boldsymbol{f}_{\mathrm{V}} \cdot \dot{\boldsymbol{x}}_{\mathrm{d}}}{\|\boldsymbol{f}_{\mathrm{V}}\| \|\dot{\boldsymbol{x}}_{\mathrm{d}}\|} \frac{\boldsymbol{f}_{\mathrm{V}}}{\|\boldsymbol{f}_{\mathrm{V}}\|} & , \text{if } -\boldsymbol{f}_{\mathrm{V}} \cdot \dot{\boldsymbol{x}}_{\mathrm{d}} > 0 \\ 0 & , \text{else} \end{cases} \tag{3.8}$$

where $\boldsymbol{f}_{\mathrm{V}} = \sum \boldsymbol{f}_{\mathrm{V}i}$ is the workspace boundary wrench generated by the virtual environment. In other words, no velocity can be generated towards the virtual boundaries.

3.4.5 Frame interpolation

The integration process ensures the continuity of the robot's desired pose when using the direct control mode. However, this continuity is not guaranteed with our shared control approach because a desired pose $\mathcal{E}_{\mathrm{SCT}}$ for the end-effector is computed at a rate of 30 Hz. This pose cannot be directly applied to the Cartesian Impedance controller, which operates at 1 kHz. To address this, we employ a frame interpolation module that achieves two objectives: first, it continuously tracks $\mathcal{E}_{\mathrm{SCT}}$ provided at a comparably low rate while sending $\mathcal{E}_{\mathrm{FRAME}}$ at 1 kHz. Second, it ensures that the resulting trajectory respects translational and rotational velocity limits, guaranteeing safety.

To achieve this, the frame interpolator performs a linear interpolation towards the current SCT desired frame $\mathcal{E}_{\mathrm{SCT}}$. While the translational component can be interpolated straightforwardly, the rotational component requires spherical linear interpolation using the quaternion representation [Sho85]. Linear

interpolation allows for a straightforward integration of a velocity limit by adjusting the interpolation time T_m. To ensure the robot does not exceed the speed of the source signal, we define the final interpolation time $T_q = \max(\mathrm{T_m}, \mathrm{T_s})$ where $T_s = 1/30\,\mathrm{Hz}$ is the sample time of the shared control module.

This piece-wise linear interpolation results in discontinuities in the velocities of the resulting trajectory. A second-order filter is applied to create a smooth trajectory that can be used as input to the Cartesian impedance controller. Weitschat *et al.* [WDV16] provides more details on the frame interpolator approach.

3.5 Shared control unit

Assisting users is possible due to the combination of multiple processes: one to coordinate the various events happening while using the system, one to infer the available tasks and the user's intent, and one to assist with shared control.

3.5.1 Coordination

An event-based finite-state machine coordinates EDAN's software to manage its various system states. These states include which device the user wants to control (such as the tablet GUI, the arm, the wheelchair, or none), the type of interpolator in use (frame, velocity, joints), the wheelchair's control mode (with or without whole-body control, and whether it handles translations or rotations), the active task, etc. Different combinations of these control modes define the high-level states that the EDAN robot can be in. These states can change based on user input, such as selecting a task from the tablet or completing a task like releasing a bottle. Additionally, processes within the system can trigger state changes; for example, the task inference can initiate shared control support for a *pick* task with the frame interpolator when the robot is near a pickable object.

3.5.2 Shared Control Templates

The SCT framework is the main contribution of this thesis. It is formalized in Chapter 4 and extensions are presented in Chapters 6 and 7. SCTs provide assistance for the user by guiding and constraining the end-effector movement during the different subtasks of a task [Que+20]. The provided assistance is independent of the user intent estimation: the assistance is provided for a single, known task. The user is always in control, as no robot motion occurs without user input, and they can stop the task at any point or switch to another task. It takes as input user commands and outputs the SCT target pose $\mathcal{E}_{\mathrm{SCT}}$.

It is interfaced with the whole-body controller via the frame interpolator, providing safety: even if the target pose is incorrect for any reason (wrong object detection, edge case in skill definition, etc), the interpolator outputs a smooth trajectory respecting velocity limits. The whole-body controller, as described in subsection 3.4.1, expands the workspace of the robot arm to allow for tasks necessitating a large range of motion, e. g. opening a door. As such, the wheelchair follows the end-effector to maintain arm manipulability as soon as the end-effector crosses geometric boundaries, which can be defined as state parameters. For example, when opening the door, the wheelchair moves forward when the arm gets out of reach, so there is no need to switch to wheelchair control. Similarly, the wheelchair moves back when the arm gets too close to the user when opening a drawer. With this approach, the user can focus on controlling the end-effector with shared control guidance. At the same time, the local commands to the robotic arm *and* the wheelchair are generated from the whole-body control, reducing the user's workload.

3.5.3 Task inference and user intent estimation

This process offers two key functions: first, it determines which shared control task can be executed based on the current state of the world; second, it computes a likelihood that a specific task corresponds to the

user intent and automatically activates support once this likelihood is high enough. We use the PDDL to define skills with specific preconditions and effects, which influence the symbolic state of the world. At a frequency of 10 Hz, the process assesses which tasks correspond to the objects currently instantiated in the world and selects those with valid preconditions. The resulting list is then displayed on the tablet GUI, ordered by likelihood.

The likelihood of starting a task is affected by two factors. The first is the Euclidean distance between the end-effector and the task target. The second is the distance between the end-effector and its projection by the active constraints of the first SCT state of a potential task. For example, an *approach* skill with a cone constraint in its first state can smoothly guide the end-effector toward a grasp frame. A task will start automatically as soon as the distance to the target is below a certain threshold and the active constraints are fulfilled to a certain level, e. g. being close enough to the cone constraints in the case of an *approach* skill.

Estimating the user intent operates independently from the shared control execution: the robot either estimates the user intent or provides assistance. The user can modify the task if the goal estimation is incorrect. This makes the robot's state transparent, but it also makes it difficult to deal with a packed scene such as a full fridge: it will be difficult to estimate the correct target with many objects close to one another in the scene. Future research will explore more sophisticated methods for calculating the likelihood of tasks, taking inspiration from existing work in the field [JA19].

If there is no clear target for a task such as placing a grasped object on a table, the task may never be automatically started. Other tasks, such as putting the manipulator in its parking position, never start automatically. In this case, the user can either send a trigger to start the most likely task (the highest one on the available task list) or switch to controlling the tablet GUI and select whichever task they want to start.

3.6 Spectrum of autonomy

The goal of our system is to provide a spectrum of autonomous capabilities for each task, giving users the flexibility to choose and adjust the level of autonomy they prefer. These preferences are also expected to change over time. In addition to the previously discussed direct and shared control methods, Bustamante *et al.* [Bus+21] proposed a method that allows for smooth transitions between shared control and supervised autonomy. When the user activates it during a task, an automaton generates commands based on a cost function designed to efficiently progress to the next state while avoiding obstacles. For example, a user might select the direction of approach to a mug and then trigger the automaton to complete the motion of picking it up.

Finally, in certain situations, the user may be unable to complete a task or may require assistance due to reasons such as a medical emergency. In these cases, telepresence control by an expert is possible [Vog+20a]. This approach allows the remote operator to experience a high level of immersion or transparency, enabling them to perceive the remote environment through their own senses as if they were physically present at the remote site. Telepresence systems typically provide visual, auditory, and haptic (kinesthetic and tactile) feedback from the remote environment. The setup may include using a manipulator as an input device, such as the DLR HUG robot (see [Vog+20a]), or a 6-DoF haptic device. It also incorporates a head-mounted display with a tracking system for head motion, sensor gloves to control the robotic fingers, and a 1-DoF haptic primary device that transmits the grasping forces of the robotic hand back to the operator. For safety, the human operator is connected to the robots via safety clutches and operates a foot-controlled dead man's switch. The EDAN robot's stereo cameras are mounted on a pan-tilt unit, allowing the operator to actively change their viewing direction using the head-mounted display's tracking system.

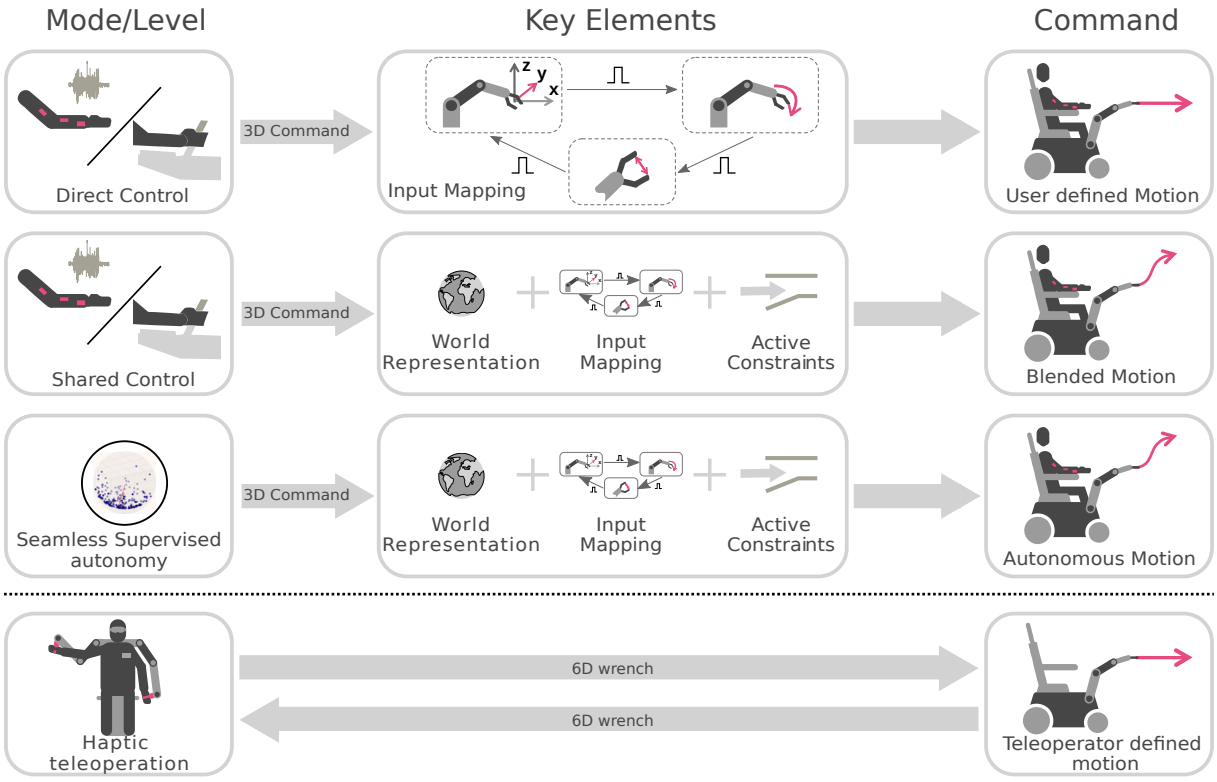


Figure 3.7: Three levels of autonomy are available with EDAN, with four means of control: direct control, shared control, seamless supervised autonomy, and haptic teleoperation.

3.7 Conclusion

We introduced the various components of the EDAN assistive robot that enable users to interact with their surroundings and perform sequences of activities of daily living. Assisting a user involves using most of the components previously discussed. The perception pipeline is combined with the anchoring to instantiate a world model, which is then used by the task inference to identify which tasks are currently feasible. A task will start once likely enough, where Shared Control Templates will support the users. Meanwhile, users can continuously send 3D commands to the robot through a joystick or an EMG-based interface. The SCT end-effector frame $\mathcal{E}_{\mathrm{SCT}}$ is then interpolated and sent to the whole-body impedance controller, which, in conjunction with virtual workspace boundaries, ensures safe interactions of the robot with the user and its environment.

The following chapter will provide a detailed overview of the design and capabilities of the Shared Control Templates framework.

Chapter 4

Shared Control Templates

The results in this chapter have been partially presented at ICRA 2020, see [Que+20].

4.1 Introduction

The purpose of our robot is to assist its users with tasks, such as opening a door or pouring a drink. Formally, a task represents "an abstract encoding of something that a robotic system is able to perform" [Lut22]. Tasks can be composed hierarchically. For example, a drinking task can be decomposed into more granular tasks: grasping a filled container, pouring its contents into a glass, placing the container back on a table, grabbing the glass, and bringing it to the user's mouth. A task does not directly link to functionalities and is, therefore, an abstract representation of how to perform something. Hence, it will hold domain knowledge such as semantic information (preconditions and effects) but is independent of the functionalities used to realize the task (e. g. whether one uses autonomy, shared control, or teleoperation).

A skill provides the realization of functionalities for a task. They are linked to concrete implementations and robot components. For example, a robot could have skills to localize an object with a neural network and a camera or plan a path with graph search to move the wheelchair from one point to another. In this work, we focus on shared control skills, which enable the robot to assist the user in realizing tasks.

We developed a novel shared control approach, Shared Control Templates (SCT), to assist users in activities of daily living in a robust and legible manner. An SCT specifies the information required to provide a robot with a shared control skill, such as *pour from a grasped object into container* or *grasp a cylindrical object*. An SCT holds references to abstract objects such as *grasped object* or *target object*. Once a task is selected based on user actions, an SCT is instantiated at runtime, and all abstract variables are bound to specific instances of objects in the world model. It takes user commands $\boldsymbol{u}(t)$ as input, $\boldsymbol{u} \in \mathbb{R}^n, n \in [\![1, 6]\!]$ and outputs the target SCT pose $\mathcal{E}_{\mathrm{SCT}}$. Due to the considered user interfaces, $n = 3$ in this work, but it might be different for other user interfaces or adjusted explicitly for a user's capabilities.

4.1.1 Mathematical notations

We follow the definitions from "A Mathematical Introduction to Robotic Manipulation" by Richard M. Murray. A summary of the notions of interest is presented here.

We are primarily interested in the robot task space, i. e. the space occupied by the end-effector of the robotic manipulator. The end-effector of a modern robot such as EDAN has 6 DoFs, 3 for its Cartesian position and 3 for its orientation. Its pose equals its localization in task space and is defined using a reference coordinate frame – a reference system used to define the position and orientation of objects in space. By default, we express poses in a coordinate frame static w. r. t. the floor, noted $\mathcal{W}_s$, set at robot bootup as the initial pose of the base $\mathcal{W}$ of the robotic manipulator, defined in Fig. 3.6.

A point A in space can be defined as $\boldsymbol{p}_A \in \mathbb{R}^3$. Orientations and rotations in task space can be represented by matrices belonging to the Special orthogonal group 3: $SO(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times3} : \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}, det(\boldsymbol{R}) = +1\}$. From this is defined the Special Euclidean group SE(3): $SE(3) = \{(\boldsymbol{p}, \boldsymbol{R}) : \boldsymbol{p} \in \mathbb{R}^3, \boldsymbol{R} \in SO(3)\} = \mathbb{R}^3 \times SO(3)$. Hence, a pose in task space can be represented as an element of SE(3). An element $(\boldsymbol{p}, \boldsymbol{R}) \in SE(3)$ can also express a rigid body transformation, the transformation from one coordinate frame to another.

One can also use the homogeneous representation of a transformation $(p_{AB}, R_{AB}) \in SE(3)$, defined as the 4×4 matrix of the form ${}^A\boldsymbol{H}_B = \begin{bmatrix} \boldsymbol{R}_{AB} & \boldsymbol{p}_{AB} \\ 0 & 1 \end{bmatrix}$. Hence, the pose of the end-effector written as a homogeneous representation expressed in $\mathcal{W}_s$ is ${}^{\mathcal{W}_s}\boldsymbol{H}_{\mathcal{E}}$ or simply $\boldsymbol{H}_{\mathcal{E}}$.

4.1.2 Core concepts

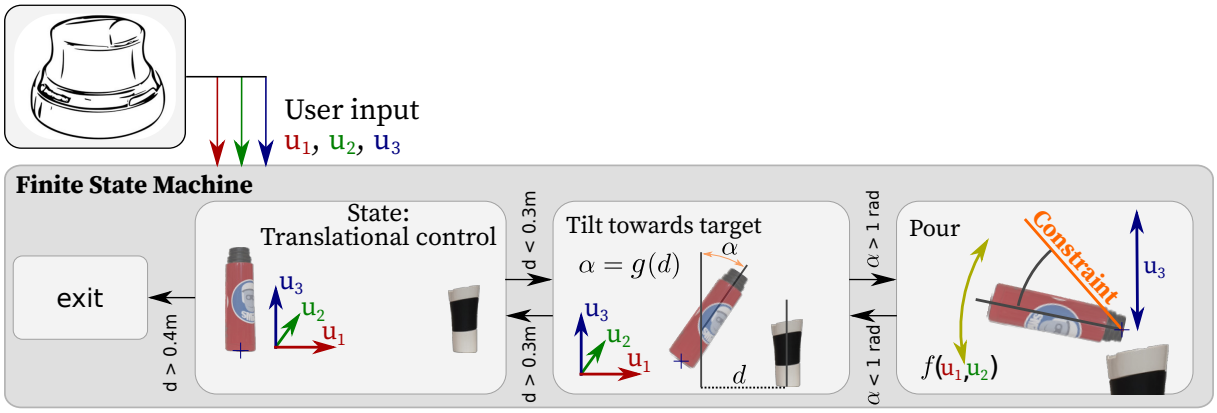


Figure 4.1: **SCT for a *pour* skill.** User inputs come from a joystick, here a Spacemouse. The skill is represented with three main states, each with its input mapping and active constraints. Transitions are based on the horizontal distance to the target (between 'Translational control' and 'Tilt towards target') and the tilt angle of the grasped object (between 'Tilt towards target' and 'Pour'). In 'Tilt towards target', the orientation is based on the position. In 'Pour', the user controls the grasped object tilt angle – with a specific input mapping IM_{pour} – and height. The user can then backtrack in the finite-state machine and exit, or pour from another angle.

An SCT encodes skills as finite-state machines, modeling the different phases of a skill as states. Formally [HU79], a deterministic finite-state machine consists of the following:

1. A finite set called the input alphabet is all possible inputs or commands the finite-state machine can receive to trigger state transitions, such as sensor data, user commands, or environmental changes, see subsection 4.4.2.
2. A finite set of states.
3. A start state which becomes active when the instantiated SCT is activated.
4. A state-transition function, listing the required predicates to go from one state to another.
5. A set of final states, in our use case consisting of a single state called 'exit'. If this state is reached, the execution of an instantiated SCT finishes.

For example, the skill *pour water* comprises four states: 'Translational control', 'Tilt towards goal', 'Pour' and 'exit', see also Fig. 4.1.

Each state can contain input mappings, active constraints, and robot parameters. An input mapping maps the low-dimensional user inputs (for example, coming from a joystick or EMG-based interface) to task-relevant motion of the end-effector target pose $\mathcal{E}_{\mathrm{SCT}}$. Active constraints additionally constrain $\mathcal{E}_{\mathrm{SCT}}$ to guide the user and restrict unsafe motions, such as tilting a filled container when moving it around or preventing it from hitting a table. Robot parameters can, for instance, relate to end-effector finger configuration, impedance control, or whole-body control. State transitions depend on distance metrics, end-effector contact forces with the environment, or user triggers. Skills are object-centric and based on frames, inspired by the task frame formalism [BD96].

SCTs are specified in human-readable *YAML* files. This makes it convenient to develop and edit skills without modifying (or knowing about) the system or control software. An example of skill specification can be seen in Listing 4.3.

4.1.3 Feature frames

An input mapping or an active constraint can be applied to various frames, such as those representing the end-effector, the center of a grasped object, or the lid of a container being held. These feature frames, denoted as F, can be static and typically represent object properties stored in the database, such as "tool frame" or "tip frame." Alternatively, custom frames can be defined, such as a frame at the tip of a grasped bottle that always points towards a target object during a *pour* task, inspired by approaches like the task frame formalism [Mas81].

A frame's position and orientation can be specified. It can be subject to a rigid body transformation. A frame can also be configured to point towards another frame, copy its orientation, or align an arbitrary axis with a reference frame's axis. When the frame involves the end-effector, various options are available, such using as the SCT target $\mathcal{E}_{\mathrm{SCT}}$, the SCT target at the start of a state $\mathcal{E}_{\mathrm{START\_OF\_STATE}}$, the interpolator target frame $\mathcal{E}_{\mathrm{FRAME}}$ (see subsection 3.4.5), or the measured end-effector pose $\mathcal{E}_{\mathrm{MEASURED}}$, see subsection 4.4.4 for their impact on transitions. These features enable the specification of a wide variety of frames.

4.2 Input mapping

4.2.1 Definition

An input mapping (IM) is a model constraining the velocity commands applied to the end-effector. It maps user actions $\boldsymbol{u}(t)$ to the target end-effector frame velocity $\dot{\boldsymbol{x}}(t)$. Discrete systems such as EDAN have a time step duration, Δt. At each time step, an input mapping maps $\boldsymbol{u}(t)$ to a small end-effector motion, i. e. a displacement, $\delta\boldsymbol{H}$, which is then applied on the target end-effector pose $\mathcal{E}_{\mathrm{SCT}}$.

Formally, at each time step, an input mapping im first computes a displacement $\delta\boldsymbol{H}$:

$$\begin{aligned} \mathrm{map}_{im} \colon \mathbb{R}^n &\to SE(3) \\ \boldsymbol{u}(t) &\mapsto \delta\boldsymbol{H} \end{aligned} \tag{4.1}$$

For example, the displacement for a translational control input mapping, see Fig. 4.2 *top*, is written as $\mathrm{map}_{im}(\boldsymbol{u}) = k_{\mathrm{trans}}\boldsymbol{u}\Delta t\boldsymbol{M}$ with k_{trans} the velocity limit in translation (see Equation 3.7) and M a matrix mapping command DoFs to motion DoFs, here a rectangular identity matrix.

This results in:

$$\delta\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & k_{\mathrm{trans}} * u_1 * \Delta t \\ 0 & 1 & 0 & k_{\mathrm{trans}} * u_2 * \Delta t \\ 0 & 0 & 1 & k_{\mathrm{trans}} * u_3 * \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Then, this is applied on the SCT target pose $\mathcal{E}_{\text{SCT}}$, resulting in a new pose $\mathcal{E}_{\text{IM}}$:

$$\begin{aligned} \text{displace}_{im} : SE(3), SE(3) &\rightarrow SE(3) \\ \boldsymbol{H}_{\mathcal{E}_{\text{SCT}}}, \delta\boldsymbol{H} &\mapsto \boldsymbol{H}_{\mathcal{E}_{\text{IM}}}(t). \end{aligned} \tag{4.3}$$

A displacement can be applied either in the local frame of reference, here the end-effector: $\text{displace}_{im}(\boldsymbol{H}_{\mathcal{E}_{\text{SCT}}}, \delta\boldsymbol{H}) = \boldsymbol{H}_{\mathcal{E}_{\text{SCT}}} * \delta\boldsymbol{H}$ or the wheelchair frame of reference (i. e. the user perspective): $\text{displace}_{im}(\boldsymbol{H}_{\mathcal{E}_{\text{SCT}}}, \delta\boldsymbol{H}) = \delta\boldsymbol{H} * \boldsymbol{H}_{\mathcal{E}_{\text{SCT}}}$. We use an Euler angle representation to compute rotational displacements, as we only compute incremental rotations.

Elementary input mappings are a 1-to-1 mapping, illustrated in Fig. 4.2, top and middle rows.

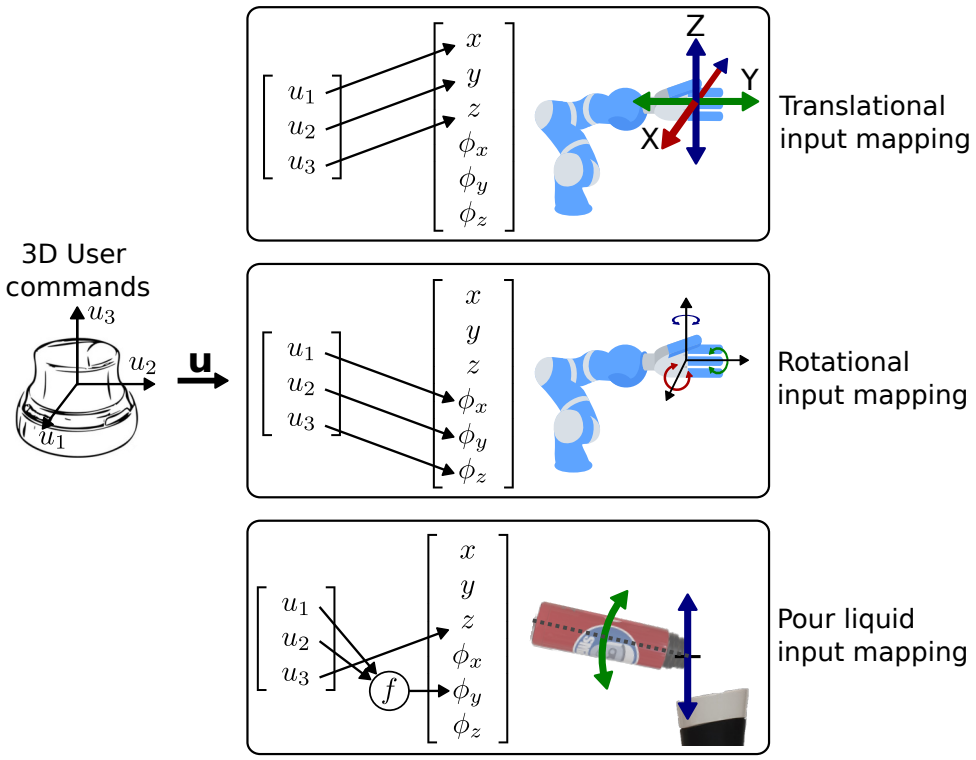


Figure 4.2: **Three input mappings.** The first and second are directly applied on $\mathcal{E}_{\text{SCT}}$, while the third is applied on the tip of a grasped bottle, with f a scalar product between a partial command vector $[u_1, u_2]$ and an axis of a frame at the tip of the bottle.

Multiple input mappings can be active simultaneously, iteratively applying their displacements on the end-effector pose. An input mapping might not be directly computed on the end-effector but can be applied on feature frames F, e. g. at the tip of a grasped bottle, as seen in Fig. 4.2, *bottom*. Algorithm 1 describes the application of such an input mapping.

4.2.2 Input mappings collection

A 1-to-1 mapping isn't always the most intuitive way to control a robot. It can be helpful to scale down commands in directions perpendicular to the primary motion, such as scaling down vertical or lateral motions when opening a drawer by pulling towards oneself. Additionally, commands might need to be scaled down in proportion to the distance from a feature frame, such as in an insertion task, to enhance precision. Implementing a deadzone can help filter out low-intensity, potentially unintentional commands. These features are particularly relevant when dealing with noisy interfaces such as EMG-based ones.

Algorithm 1 1-to-1 input mapping with feature frame

Input: User input $\boldsymbol{u}$, $\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1)$, Input Mapping IM
Output: Displaced target end-effector pose $\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}(t)$
$F \leftarrow$ IM.compute_feature_frame($\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1)$)
// Compute transform from $\boldsymbol{H}_F$ to $\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1)$
${}^{F}\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}} \leftarrow {\boldsymbol{H}_F}^{-1}\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1)$
// Compute displacement
$\delta\boldsymbol{H} \leftarrow$ euler_to_homogeneous_transformation($\boldsymbol{u}$)
// If Local: Apply local displacement to the feature frame
$\boldsymbol{H}_{F_{\mathrm{DISPLACED}}} \leftarrow \boldsymbol{H}_F\delta\boldsymbol{H}$
// Update target end-effector pose from the new feature frame
$\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}(t) \leftarrow \boldsymbol{H}_{F_{\mathrm{DISPLACED}}}{}^{F}\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}$
return $\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}(t)$

More complex functions can also be beneficial, especially when controlling rotations, such as during a pouring task. In the current implementation of the *pour* skill, two DoFs of the input mapping, u_1 and u_2, are used to control the rotation around the tip of a grasped bottle. These DoFs correspond to movements in the $\boldsymbol{x}$-$\boldsymbol{y}$ plane during translational control. The rotational displacement is calculated as the scalar product between the partial command vector $[u_1, u_2, 0]$ and the frame at the bottle's tip, where the $\boldsymbol{x}$-axis points toward the target, and the $\boldsymbol{z}$-axis is vertical.

This approach provides intuitive control for the user. As illustrated in Fig. 4.1, during states 'Approach' and 'Tilt toward target', the user directs $[u_1, u_2]$ towards the target to move closer, with orientation governed by an active constraint. However, the input mapping shifts in the state 'Pour,' and the user now controls the rotation. The user must issue the same commands directed toward the target to continue tilting. In practice, the user remains unaware of the change in input mapping (or even the state change). By maintaining consistent commands, the user can seamlessly progress towards completing the task, pouring into the target container.

It's important to note that this method grants the user control over the primary DoFs of the task, allowing them to choose the angle of approach (from the side, from behind) and to decide how much, how quickly, and at what height to pour. However, they cannot aim anywhere other than the center of the target container – unless they intentionally enter a correction state using a trigger, see section 4.6.

In summary, an input mapping can be tailored to the desired level of autonomy and the finesse of the user input. It can vary from following a one-dimensional virtual guide to allowing detailed, multi-DoFs control.

4.2.3 Velocity limits

It is helpful to subject the end-pose $\mathcal{E}_{\mathrm{IM}}$ resulting from input mappings to velocity limits to not go too far from the interpolator target frame $\mathcal{E}_{\mathrm{FRAME}}$. As a user observes $\mathcal{E}_{\mathrm{MEASURED}}$ but their commands are applied to $\mathcal{E}_{\mathrm{SCT}}$, transparency is lost if $\mathcal{E}_{\mathrm{MEASURED}}$ and $\mathcal{E}_{\mathrm{SCT}}$ diverge. Velocity limits are also relevant when transitioning from state to state, described in detail in subsection 4.4.4. Therefore, we enforce that $\mathcal{E}_{\mathrm{IM}}$ is subject to the same velocity limits as in velocity control, see Equation 3.7.

4.3 Active constraints

4.3.1 Definition

Active constraints (AC) restrict the task space and guide the user along a task. Complementary to an input mapping, which applies to velocities, an active constraint affects the position of a frame. We use active constraints as regional constraints as defined by Bowyer *et al.* [BDB13], which keep a desired end-effector pose within a restricted region of the task space. In this work, this is achieved by projecting the target end-effector pose (or any other frame of interest) back into the allowed region if constraints are violated.

For an active constraint ac:

$$\begin{aligned} \text{project}_{ac}\colon SE(3) &\to SE(3) \\ \boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}(t) &\mapsto \boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}(t) \end{aligned} \tag{4.4}$$

Unlike *force constraints* in haptic teleoperation, which apply increasing forces as the user violates them, active constraints cannot be overcome by the user. As with input mappings, an active constraint can be applied on feature frames F. Changes on F are then reflected on a new $\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}$ pose, cf Algorithm 2.

Algorithm 2 Active Constraint

Input: $\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}(t)$, Active Constraints ACs
Output: Constrained target end-effector pose $\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}(t)$

$\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}} \leftarrow \boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}(t)$
for AC in ACs **do**
  $\boldsymbol{H}_F \leftarrow$ AC.compute_feature_frame($\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}$)
  ${}^{F}\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}} \leftarrow {\boldsymbol{H}_F}^{-1}\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}$
  // Apply constraints on the feature frame
  $\boldsymbol{H}_{F_{\mathrm{PROJECTED}}} \leftarrow \text{project}_{AC}(\boldsymbol{H}_F)$
  // Update target end-effector pose from feature frame
  $\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}} \leftarrow \boldsymbol{H}_{F_{\mathrm{PROJECTED}}}{}^{F}\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}$
end for
return $\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}$

Multiple active constraints can be applied iteratively within the same state. In contrast to other approaches in the literature [AD14], our method does not include a mechanism for detecting conflicting constraints. However, any model featuring a projection function onto SE(3) can be used, providing flexibility in defining constraints. The counterpart is that careful design is required from the skill developer.

4.3.2 Constraint definition

The simplest type of constraint involves fixing a specific DoF to a constant value expressed in the relevant reference frame. For example, when pushing open a door, the end-effector might be constrained to maintain a specific height w. r. t. the door handle (see Fig. 4.3). Constraints can also be defined using functions that compute DoF values involving inequalities, polynomials, additions, scalings, and dot products. These functions take inputs such as distance metrics and feature frames; for instance, in Fig. 4.1, the state 'Tilt towards target' is defined by a function where the angle α depends on the distance between the end-effector and the target. An orientation constraint can also be based on the end-effector position and possibly learned from demonstration-based approaches (see chapter 7).

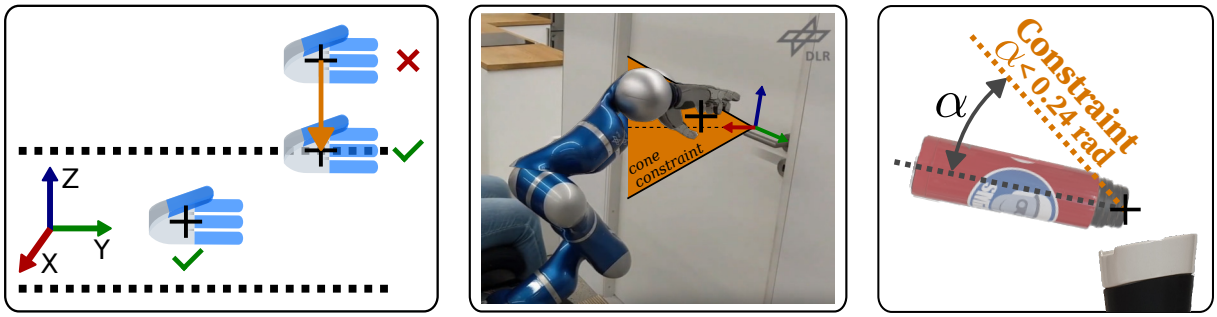
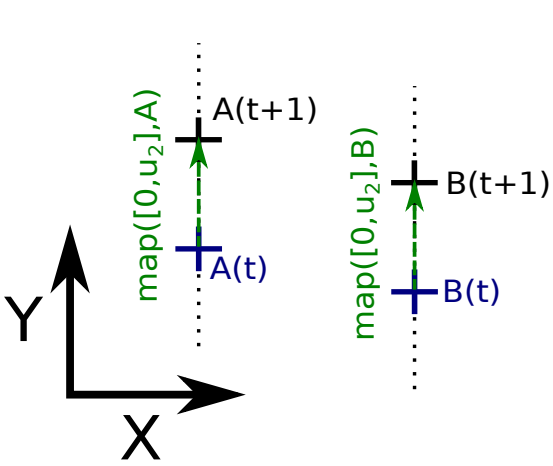
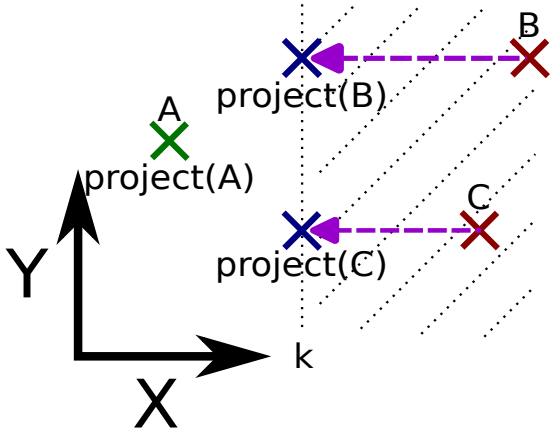


Figure 4.3: **Active constraints examples. Left**: End-effector constrained to stay within a specific range in height. **Center**: End-effector constrained within a cone. As the user gives forward commands, the end-effector is guided toward a grasp pose. **Right**: Upper limit on the orientation of the tip of a bottle, to not spill over.



(a) The covered task space for an **input mapping** such that $\delta \boldsymbol{H} = [0, u_2]$ depends on the initial position when the state activates.



(b) The available task space for an **active constraint** such that $x < k$ is independent from the initial position. A is within the regional constraints, while B and C are outside and get projected.

Figure 4.4: Illustrative example of the subspace covered by an input mapping and active constraint in a 2D space.

Position constraints can be defined using 3D Euclidean objects, which are easy to visualize. The end-effector can be constrained to project onto the surface of these objects, remain within their volume, or stay outside of it. Lines, planes, cylinders, cones, funnels, cubes, and generalized cylinders are examples of such objects. Notably, the projection doesn't have to be onto the nearest point on the surface. For example, projecting onto a cone along the axis perpendicular to its normal creates a "sliding" effect towards the cone tip, rather than a "sluggish" approach, where being in contact or not does not affect the speed of the approach.

As with input mappings, velocity limits are enforced on $\mathcal{E}_{\text{AC}}$, following Equation 3.7.

4.3.3 Complementarity between input mappings and active constraints

Fig. 4.4 shows that, in the case of an input mapping, the accessible task space varies based on the initial pose of the end-effector. However, the available task space for an active constraint remains unaffected by the initial end-effector pose.

Some constraints can be modeled either through an input mapping or as an active constraint, such

as when moving along a circle. If the end-effector is already on the circle at time t_0, an input mapping can be defined as the tangent direction along this circle. With an ideal discretization, this will restrict the user to only move along this circle. Alternatively, one could allow 2D translational motion with the input mapping and then use an active constraint to project $\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}$ onto the circle. In both cases, the resulting set of possible end-effector poses remains the same, meaning there is no single template definition for the same task space coverage. The decision to use input mappings or active constraints typically hinges on the desired behavior, ease of use (it affects the required commands to get the desired behavior), and the complexity involved in defining the SCT.

4.4 Finite-state machine

4.4.1 State components

A state is a structure optionally composed of input mappings and active constraints. Additionally, while Shared Control Templates and its modules are robot-agnostic, a successful realization will also require defining robot-specific parameters. State parameters refer to robot settings that can be relevant for task execution. These include:

- Desired finger joint position and stiffness.
- Torque safety levels: these thresholds dictate when EDAN switches to safety mode (gravity compensation control) due to external forces; see subsection 3.4.3. By default, the threshold is set comparatively low but is increased for tasks requiring significant interaction with the environment, like opening doors or drawers.
- Stiffness parameters for the impedance control for position and orientation of the end-effector, i. e. $\boldsymbol{K}_x$, see Equation 3.2.
- Whole-body control mode and boundary values: these settings determine when the mode is active and whether it allows translational motions of the wheelchair, rotational motions, or both, see Fig. 3.6.
- Elbow target point and associated stiffness: specify a null space attractor in terms of a virtual spring attracting the elbow of the manipulator via the control action $\boldsymbol{\tau}_{\mathrm{null}}$, see Equation 3.5. This is used for example to facilitate passing through a door or to keep the elbow away from a surface.
- Desired angle for the first axis: added as a low-priority task to the hierarchical impedance controller for stability.

Additionally, a range for the maximum allowable force can be defined, specifying the upper limits of force applied to the environment in each Cartesian direction ($\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ in robot base coordinates, both positive and negative). When these thresholds are reached, the latest target end-effector frame ($\mathcal{E}_{\mathrm{SCT}}$) is projected onto the plane defined by the last valid SCT target position and the normal direction in which the robot would otherwise exceed the force limit. This concept is similar to how velocity commands interact with virtual walls, as discussed in Section 3.4.4 and Equation 3.8.

This approach ensures that excessive force is avoided in undesired directions during tasks where the manipulator interacts with the environment. For instance, when pulling a drawer handle, enough force should be applied to open the drawer, but not so much that the fingers could be damaged once the drawer reaches its fully open position and can no longer move.

4.4.2 Event types for the finite-state machine

Events from the input alphabet can be categorized in various ways. On of those are spatial conditions. These occur when a specific metric, such as the Euclidean distance between the end-effector and a target

object, crosses a predefined threshold. For instance, in the *Pour* skill illustrated in Fig. 4.1, a transition from 'Translational Control' to 'Tilt Towards Target' is triggered when the horizontal distance between the grasped thermos and the target mug drops below a specific value, in this case, 0.3m. Its definition in the skill YAML file is shown in Listing 4.1. The predicates return True when the value being evaluated is outside the range, which is in practice a more convenient behavior than the opposite. Another example is when vertical movement is required after an object is grasped. There, the vertical distance between the current end-effector position and its position at the start of the state can be used as the metric, with a transition predicate set for when a displacement of 0.1m is reached.

```
predicates:
- aux_fct:
    function: EuclideanDistance
    frame: EE
    reference: target
    mapping: [1,1,0,0,0,0]
    range: [0.3, inf]
```

Listing 4.1: Definition of the transition from 'Translational Control' to 'Tilt Towards Target', based on horizontal Euclidean distance (assuming an upright target). The predicates evaluates to True when the distance is outside the range, here lower than 0.3m.

Another useful metric is the wrench exerted by the environment on the end-effector, which generally results from the wrench applied by the end-effector on the environment and varies smoothly during contact due to impedance control. This smooth variation allows for reliable detection of environmental contact, such as when releasing an object. Detecting forces to release a grasp is more reliable than using visual perception (where all surfaces might not be precisely localized, such as a shiny heating plate). Certain tasks require applying sufficient force to the environment, such as pressing a microwave button or door handle, or maintaining pressure to stay in contact with a drawer handle while opening it. Although EDAN lacks a wrist wrench sensor, it estimates the wrench using joint torque sensing and the Jacobian matrix. However, even with precise calibration, force estimation is never entirely accurate when not in contact with the environment, typically fluctuating within a range of [-3,3]N. Consequently, transitions can be based either on the absolute wrench estimate or the measured wrench compared to its value at the start of the state.

```
predicates:
- aux_fct:
    wrench: EE
    mapping : [0,0,1,0,0,0]
    range: [-inf, 8]
```

Listing 4.2: Wrench transition. If the force applied on the end-effector pushing downwards in $-z$ direction is higher than 8N, the transition predicate is True.

End-effector sensor values can also be used. For instance, the finger position and the torque exerted by the end-effector after attempting a grasp can indicate whether the grasp was successful. Additionally, time-of-flight sensors embedded in the CLASH hand can detect when a target object is within grasping range.

Temporal conditions can also serve as triggers for events. However, these are generally avoided to ensure consistent assistive behavior that is not dependent on the timing of user commands, allowing the user to complete tasks at their own pace.

External triggers can also be employed to initiate transitions. Typically, users have access to a binary trigger, which can be activated using EMG-based sensors or an external button, such as one held in the other hand. This trigger is primarily used when automatic transitions are difficult to specify, to open/close the gripper or to enter a correction mode, see section 4.6. For instance, during the 2023 CYBATHLON

challenges (see section 5.3), our pilot could control the end-effector in 3D translational mode to select which apple to pick, then trigger the gripper to close. If the grasp failed, another trigger would reopen the fingers, transitioning the finite-state machine back to the approach state and allowing the pilot to attempt the grasp again. This is defined in Listing 4.3.

```
states:
 approach:
  parameters:
   EE_config: {entity: target, config_name: pre_grasp}
  transitions:
  - to: close_gripper
    predicates:
    - trigger: True
  input_mapping:
  - mapping: [tx,ty,tz,0,0,0]
  active_constraints:
  # Euler representation for orientation.
  - mapping: [x,x,x,3.12, -0.1, 1.38] # x means the DoF is not constrained.

 close_gripper:
  parameters:
   EE_config: {entity: target, config_name: grasp}
  transitions:
  - to: grasp_tried
    predicates:
    - timeout: 0.5

 grasp_tried:
  transitions:
  - to: lift_apple
    predicates:
    - aux_fct:
      frame: EE
      reference: {origin_frame: EE.START_OF_STATE}
      function: EuclideanDistance
      mapping : [0,0,1,0,0,0]
      range: [-inf, 0.1]
  - to: reposition
    predicates:
    - trigger: True
  input_mapping:
  - mapping: [tx,ty,tz,0,0,0]

 reposition:
  parameters:
   EE_config: {entity: target, config_name: pre_grasp}
  transitions:
  - to: close_gripper
    predicates:
    - trigger: True
  input_mapping:
  - mapping: [.5tx,.5ty,.5tz,0,0,0]

 lift_apple:
  ...
```

Listing 4.3: Pick apple, from the 2023 CYBATHLON Challenges. User trigger is used to close or open the end-effector, allowing retry if the first grasp failed.

Other external triggers might involve reaching specific points along a planned trajectory for the wheelchair or robotic arm. For tasks such as opening and passing through a door, a combination of wheelchair and arm control ensures smooth operation. The wheelchair follows a planned trajectory, with the user controlling forward and backward motion along this path. When the user operates the arm, whole-body control moves the wheelchair along the trajectory. When the wheelchair is controlled, reaching a via-point on the wheelchair trajectory can serve as an event that returns manipulator control to the user.

Multiple events might be required for a transition to happen. Multiple transitions with different conditions are also possible; see Listing 4.4 for an example.

```
transitions:
- to: released_object
  predicates:
  - aux_fct:
      wrench: EE
      mapping : [0,0,1,0,0,0]
      range: [-inf, 8]
- to: released_object
  predicates:
  - trigger: True
  - timeout: 2
```

Listing 4.4: **Object release skill.** Two possible triggers: **A**) either the end-effector applies 8N on the surface (through the grasped object), or **B**) the user triggers the release and 2 seconds have passed since the entry to the current state

The finite-state machine requires careful design. Otherwise, in a skill with two states, A and B, where two transitions, A->B and B->A, are both True, the active state will oscillate between both states. The various states and transitions are defined in the same *YAML* file as the input mappings and active constraints. Object-related information such as grasp poses and end-effector grasp configurations are stored separately in the object database, allowing a skill to remain more general, see subsection 3.2.1. For example, the *pick* skill for the Spice task of the 2024 Cybathlon belongs to the abstract _spice_container class and is used for both the oil container and the salt container, but there is a specific grasp configuration are for each container.

4.4.3 Hierarchical finite-state machine

The inheritance structure of the object database allows for generalization across different object classes. Moreover, there can be some overlap in behavior among various skills. For instance, the assistive behavior during an approach phase might be the same for opening a drawer and a door, but the actions after grasping will differ. It is possible to import a state from one skill into another and from one skill into another to enhance modularity and minimize code duplication.

4.4.3.1 State import

Let's first consider importing a state A into a new skill consisting of states B and state C, with a transition from B to C. When we import state A into state B, the parameters, input mappings, and active constraints from state A are adopted as defaults for state B. Specifying a parameter in state B will add it to the parameters list and override any value previously defined by state A. Specifying an input mapping in state B will replace any input mapping pre-set in state A, as combining input mappings usually is not desirable. Specifying active constraints in state B will layer them on top of the constraints defined by state A, meaning that the constraints from state A are applied first, followed by those in state B. Therefore,

any specifications made in state B take precedence over those inherited from state A. An example of this import process is illustrated in Fig. 4.5.

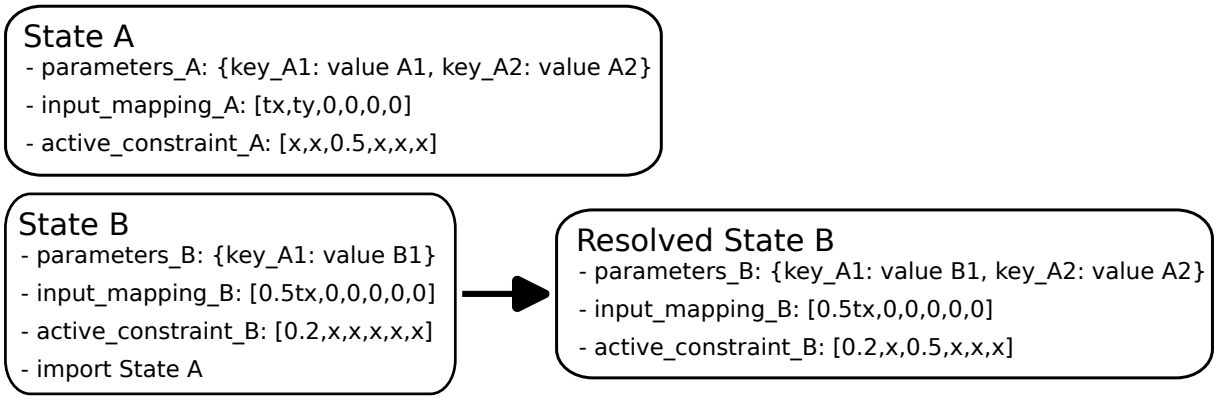


Figure 4.5: **State import example.** By importing State A, State B uses State A as the default definition.

4.4.3.2 Skill import

Instead of simply importing a state, it is also possible to import a complete skill. Let's consider a sub-skill, *approach*:

```
states:
 orient:
  parameters:
   EE_config: {entity: target, config_name: pre_grasp}
  transitions:
  - to: approach
    predicates:
    - aux_fct:
       frame: EE
       reference: target.pre_grasp_frame
       function: EuclideanDistance
       mapping : [1,1,1,0,0,0]
       range: [-inf, 0.001]
  input_mapping:
  - mapping: [tx,ty,tz,0,0,0]
  active_constraints:
  - reference: target.pre_grasp_frame
    frame: EE
    mapping: [x,x,x,0,0,0]
  - explicit_model:
     name: Cone
     frame: EE
     kwargs:
      point: {origin_frame: target.pre_grasp_frame, axis: position}
      direction: {origin_frame: target.grasp_frame_down, axis: -Z}
      aperture_angle: 0.4
      length: 0.3

 go_closer:
  transitions:
  - to: proceed
    predicates:
    - aux_fct:
       frame: EE
```

```
35        reference: target.grasp_frame
36        function: EuclideanDistance
37        mapping : [1,1,1,0,0,0]
38        range: [-inf, 0.001]
39    input_mapping:
40    - mapping: [tx,ty,tz,0,0,0]
41    active_constraints:
42    - explicit_model:
43       name: Line
44       frame: EE
45       kwargs:
46        point: {origin_frame: target.pregrasp_frame, axis: position}
47        direction: {origin_frame: target.pregrasp_frame, orientation_to_frame:
       target.grasp_frame, axis: X}
48
49  proceed
```

Listing 4.5: *approach* sub-skill.

It consists of a first state 'orient', which constrains the end-effector within a cone (lines 20-27) and aligns it with a specific orientation set by the pre-grasp frame (lines 17-19). This guides the user towards the tip of the cone, with its position defined by the pre-grasp frame. The finite-state machine transitions to the 'go closer' state once the end-effector is within a millimeter of the pre-grasp frame (lines 6-13). In the state 'go closer', only linear motion is permitted (lines 41-47) until the end-effector reaches the grasp frame. At that point, the sub-skill concludes, and the finite-state machine is ready to proceed.

This approach behavior can be applied to various tasks, such as interacting with articulated objects like opening a door or fridge or picking up an object. Constraining the approach this way is similar to the concept of capture envelopes [Mue+17]. As a result, these skills can incorporate the *approach* sub-skill, providing modularity and minimizing code duplication. This also encourages consistent assistance behavior, enabling the user to develop a reliable understanding of the robot's actions and fostering trust.

To explain the workings of this import feature, let's revisit a simple skill composed of two states: A and B. When importing *approach* into state A, the finite-state machine is modified: three new states – A.orient, A.approach, and A.proceed – are created by incorporating the corresponding orient, approach, and proceed states into state A, while the original state A is removed. As with importing a single state into another, state A can specify its parameters, input mappings, and active constraints. The transitions originally defined from A to B are now inserted as transitions from each new state – state A.orient, state A.approach, state A.proceed – to state B. Additionally, transitions can be defined within state A from any of the new states (A.orient, A.approach, A.proceed) to any other state (A.orient, A.approach, A.proceed, B). In summary, our theoretical hierarchical finite-state machine is flattened into a single-layer finite-state machine with newly defined states. This process is illustrated in Fig. 4.6.

4.4.4 Velocity limits for the end-effector target frame

The decoupling between the SCT target and the controller introduces the need to consider three distinct frames: the SCT target frame $\mathcal{E}_{\mathrm{SCT}}$, which is tracked by the interpolator frame $\mathcal{E}_{\mathrm{FRAME}}$, and itself followed by the actual robot pose $\mathcal{E}_{\mathrm{MEASURED}}$ through the whole-body impedance controller.

User safety is already ensured by the virtual workspace boundaries and the interpolator's imposed velocity limits; see subsection 3.4.2 and subsection 3.4.4, respectively. However, it is also beneficial for $\mathcal{E}_{\mathrm{SCT}}$ to comply with these velocity limits. As users observe $\mathcal{E}_{\mathrm{MEASURED}}$ but their commands are applied to $\mathcal{E}_{\mathrm{SCT}}$, transparency is lost if $\mathcal{E}_{\mathrm{MEASURED}}$ and $\mathcal{E}_{\mathrm{SCT}}$ diverge; additionally, constraints might become unsuited.

Any of those frames can be used to compute transitions, each leading to different outcomes. If $\mathcal{E}_{\mathrm{MEASURED}}$ is used, some transitions may never occur due to an offset between $\mathcal{E}_{\mathrm{SCT}}$ and $\mathcal{E}_{\mathrm{MEASURED}}$

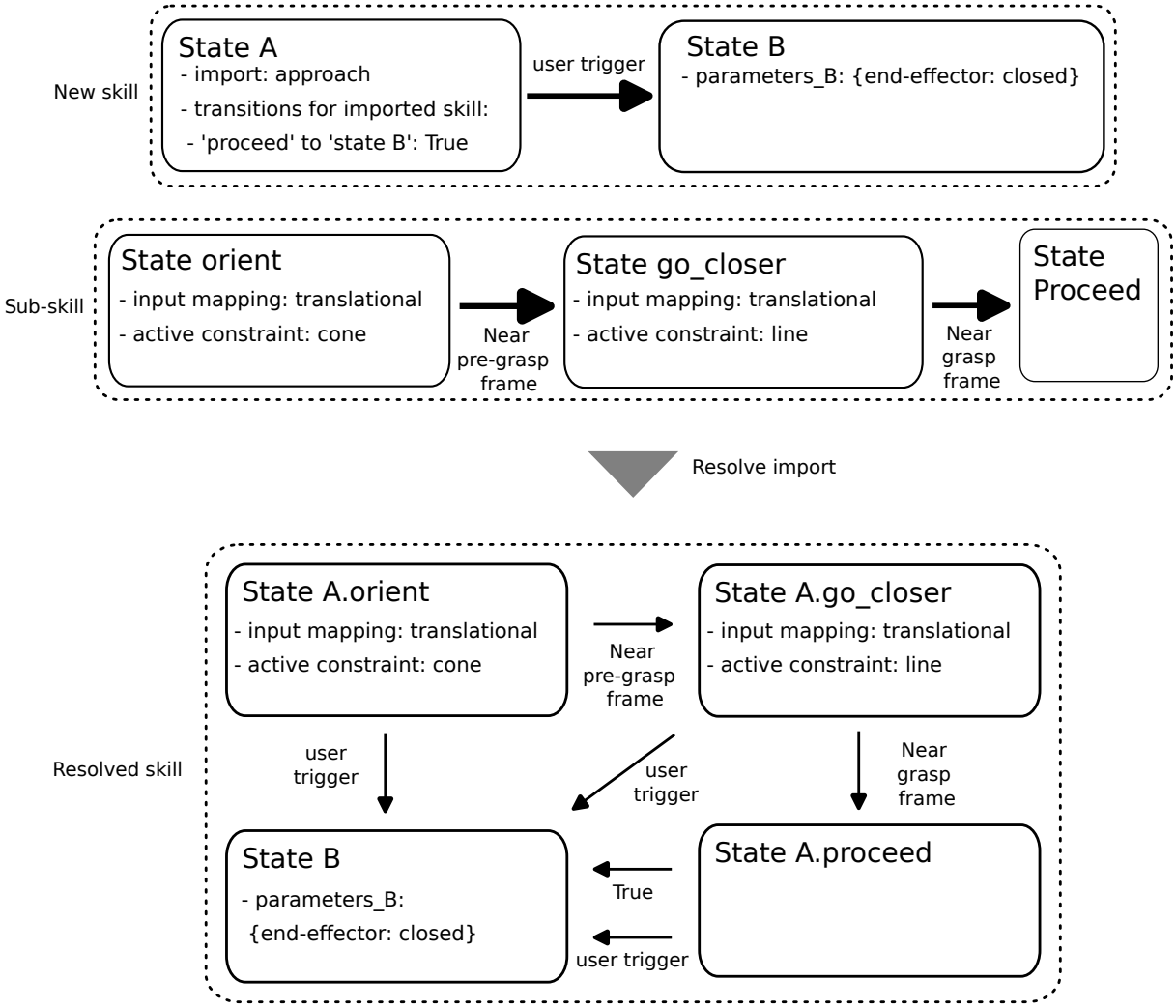


Figure 4.6: **Sub-skill import example** This example illustrates a new skill importing a sub-skill (*approach*) after flattening the finite-state machine.

when in contact with the environment or in kinematic limits. As a result, $\mathcal{E}_{\mathrm{MEASURED}}$ is used sparingly for specific situations, such as requiring the arm to be in specific regions of the task space.

If $\mathcal{E}_{\mathrm{SCT}}$ were used, there would be no guarantee that constraints – such as a specific orientation – would be enforced before transitioning from one state to another. This is illustrated in Fig. 4.7, *left*, with two states, A and B. The user can control the position of $\mathcal{E}_{\mathrm{SCT}}$ by issuing translational velocity commands, aiming to move towards state B. In state A, the assistive system imposes a specific orientation $\theta_{desired}$ on $\mathcal{E}_{\mathrm{SCT}}$, represented by the orientation of the blue arrow. At a particular timestep $t+1$, $\mathcal{E}_{\mathrm{FRAME}}$ crosses the boundary between states A and B (red arrow), triggering the transition condition and making state B the active state, allowing the task to proceed. The issue with this scenario is that the end-effector's orientation in state B may vary depending on the magnitude of the user commands. As illustrated in Fig. 4.7, *right*, once in state B, the orientation of $\mathcal{E}_{\mathrm{FRAME}}$ (red arrows) depends on the time spent in state A, sometimes resulting in undesirable behaviors.

Therefore, $\mathcal{E}_{\mathrm{FRAME}}$ should be used. In this case, one should determine how to control $\mathcal{E}_{\mathrm{SCT}}$ while waiting for $\mathcal{E}_{\mathrm{FRAME}}$ to 'catch up'. The goal is to ensure that constraints are applied to the end-effector when transitioning from one state to another, guaranteeing consistent assistive behavior across multiple

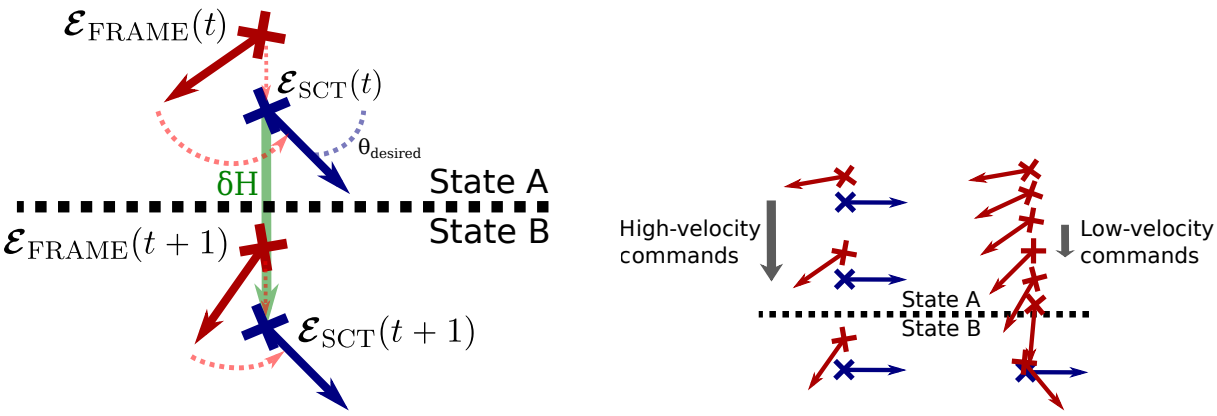


Figure 4.7: **Transition predicate based on $\mathcal{E}_{\mathrm{SCT}}$. Left**: $\mathcal{E}_{\mathrm{FRAME}}$ is tracking $\mathcal{E}_{\mathrm{SCT}}$. An orientation constraint, $\theta_{desired}$, is applied in state A on $\mathcal{E}_{\mathrm{SCT}}$. **Right**: Depending on the time spent in A, constraints may not be fully enforced when state B becomes active, affecting the assistance.

executions of the same skill while minimizing restrictions on user control. This is achieved through two features, explained below: blocking the state transition until the desired active constraints are satisfied and permitting only motions that keep the robot within state A.

4.4.4.1 Blocking the state transition with the interpolator transition condition

Let's consider the same scenario where the user gives commands to transition to state B. One should ensure that $\mathcal{E}_{\mathrm{FRAME}}$ achieves the desired orientation $\theta_{desired}$ before state B becomes active. This is accomplished by blocking the state change until $\theta_{desired}$ is reached.

Assuming the user gives commands towards state B, at a specific timestep t+1, $\mathcal{E}_{\mathrm{SCT}}$ crosses the border between states A and B, causing the transition predicate to return True. This predicate will revert to False either when B becomes the active state or if $\mathcal{E}_{\mathrm{SCT}}$ is controlled back across the boundary between the states. While the transition predicate remains True, a target end-effector pose, $\mathcal{E}_{\mathrm{SCT\_no\_velocity\_limits}}$, is computed without velocity limits. Consequently, the orientation of $\mathcal{E}_{\mathrm{SCT\_no\_velocity\_limits}}$ immediately aligns with $\theta_{desired}$, whereas $\mathcal{E}_{\mathrm{SCT}}$ will require additional timesteps to achieve this orientation. This is illustrated in Fig. 4.8. The state change is blocked until the distance (calculated in SE(3)) between $\mathcal{E}_{\mathrm{SCT\_no\_velocity\_limits}}$ and $\mathcal{E}_{\mathrm{FRAME}}$ falls below a defined threshold, named the interpolator transition condition. While the finite-state machine waits for this condition, user commands are set to 0, preventing $\mathcal{E}_{\mathrm{SCT}}$ from moving further into state B, which could disrupt the assistive behavior. Once the interpolator transition condition is met, and if the transition predicate still returns True, state B becomes active.

4.4.4.2 Keeping user agency with restricted motions

Blocking the state transition ensures motion consistency, but the assistive behavior can be further improved. While the transition predicate from state A to state B returns True, and the system waits for the constraints to be enforced and the interpolator transition condition to be satisfied, user agency can still be maintained. Instead of negating all user commands, the assistive system can first compute a candidate position, $\mathcal{E}_{\mathrm{SCT\_candidate}}$, which is identical to $\mathcal{E}_{\mathrm{SCT}}$ but not sent the frame interpolator. The transition predicate is then re-evaluated using $\mathcal{E}_{\mathrm{SCT\_candidate}}$. If the predicate returns True, $\mathcal{E}_{\mathrm{SCT\_candidate}}$ is discarded (represented by the red arrows in Fig. 4.9). If the predicate returns False, $\mathcal{E}_{\mathrm{SCT}} = \mathcal{E}_{\mathrm{SCT\_candidate}}$. The user can then move the end-effector freely until the transition predicate evaluates to True again.

In practice, a finite-state machine transition only providing a boolean evaluation can derive a behavior akin to a constraint enforced on the end-effector as long as the interpolator transition condition is not

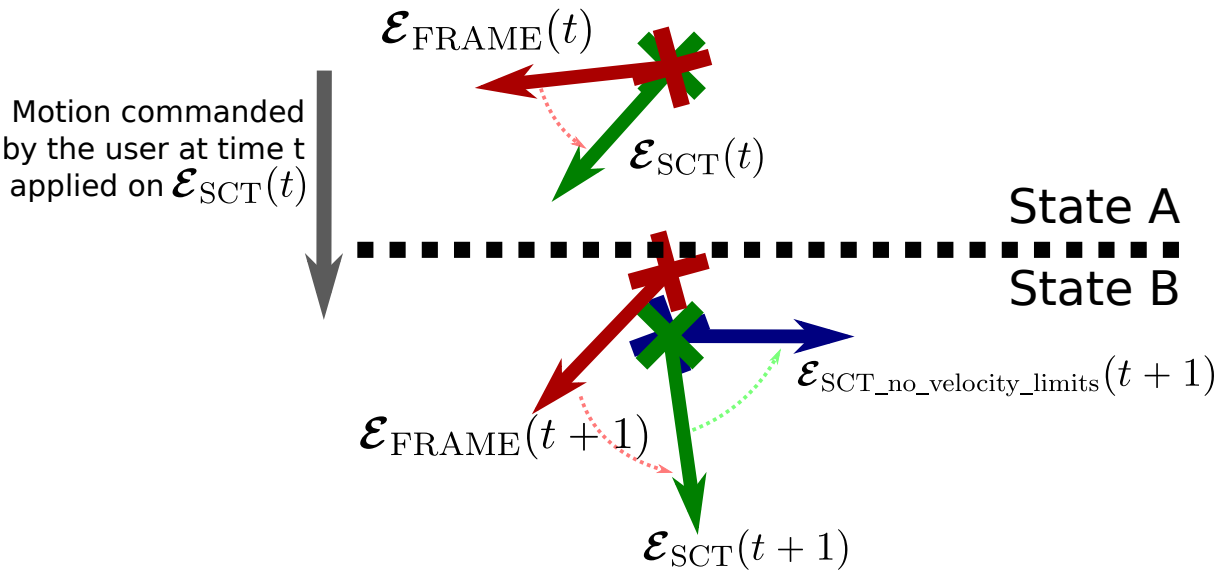


Figure 4.8: Computation of an SCT end-effector target pose without velocity limits $\mathcal{E}_{\mathrm{SCT\_no\_velocity\_limits}}$ when all predicates to change to another state are True.

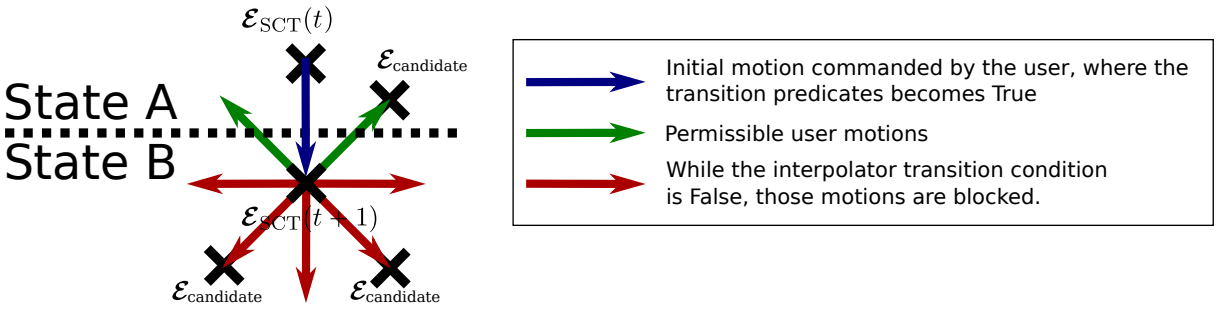


Figure 4.9: **Illustration of the effect of different commands while waiting for the interpolator transition condition to evaluate as True.** A command resulting in the transition predicate from A to B still evaluating to True (red arrow) will be discarded. However, a command taking $\mathcal{E}_{\mathrm{SCT}}$ back to state A (green arrow) will be applied to provide agency.

satisfied. In this example, commands given towards B are canceled, while commands given towards A are applied, akin to the behavior of a wall constraint.

One can also use a finite-state machine where transitions explicitly include a projection function to the transition manifold. Then, for a given $\mathcal{E}_{\mathrm{SCT\_candidate}}$, if the transition predicate evaluates to True, $\mathcal{E}_{\mathrm{SCT\_candidate}}$ can be projected back onto the transition manifold and subsequently used as the new $\mathcal{E}_{\mathrm{SCT}}$.

4.5 Instantiated SCT execution

This section presents the data flow from user command $\boldsymbol{u}$ and current SCT target pose $\mathcal{E}_{\mathrm{SCT}}$ to the robot's motion. For an active state q, we define $step_q$ as a succession of operations (here with a single input mapping and active constraint for clarity):

- Robot status and user commands are read. The end-effector target pose is $\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1)$.
- The end-effector displacement $\delta\boldsymbol{H}$ is computed: $\delta\boldsymbol{H} = \mathrm{map}(\boldsymbol{u}(t))$, see Equation 4.1.

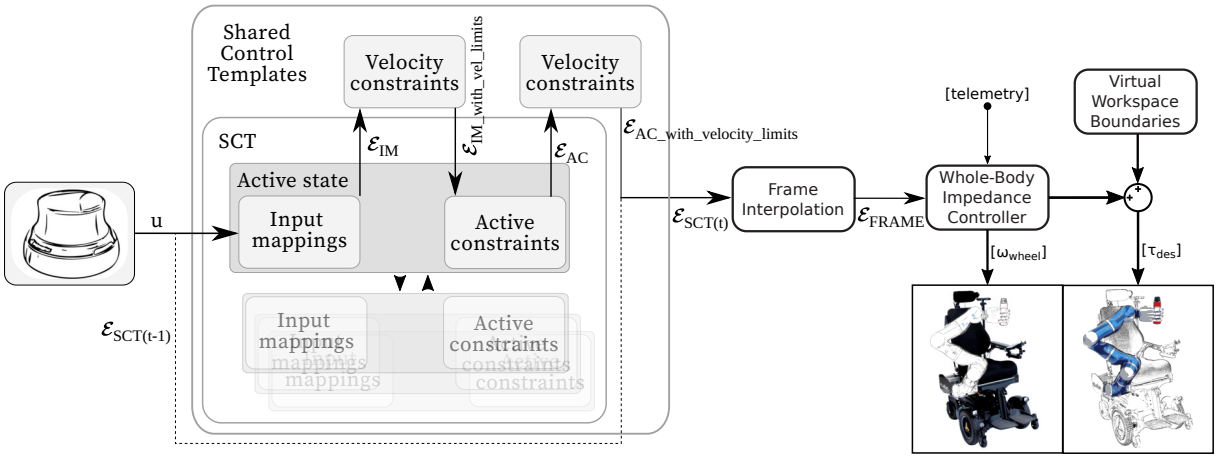


Figure 4.10: **Pipeline from user commands to robot motion through assistance with Shared Control Templates.** Based on the current *active state*, user commands move the target end-effector frame $\mathcal{E}_{\mathrm{SCT}}$ according to *input mappings*. Next, *active constraints* restrict its pose, with both mappings and constraints subject to velocity limits. The resulting target pose is then interpolated, and the output is transmitted to the whole-body impedance controller.

- This displacement is applied on the end-effector: $\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}(t) = \mathrm{displace}(\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1), \delta\boldsymbol{H})$, see Equation 4.3.
- $\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}$ is updated to satisfy velocity constraints, i. e. pulled back as close to $\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1)$ as necessary: $\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM\_with\_vel\_limits}}} = \mathrm{apply\_velocity\_limits}(\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM}}}, \boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1))$.
- The active constraint is then applied: $\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}(t) = \mathrm{project}(\boldsymbol{H}_{\mathcal{E}_{\mathrm{IM\_with\_vel\_limits}}}(t))$, see Equation 4.4.
- The new end-effector target pose is obtained by one again applying velocity limits $\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t) = \mathrm{apply\_velocity\_limits}(\boldsymbol{H}_{\mathcal{E}_{\mathrm{AC}}}, \boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t-1))$.

After computing $step_q$, $\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t)$ is sent to a pose interpolator, which creates a pose trajectory sent to the whole-body impedance controller. $\boldsymbol{H}_{\mathcal{E}_{\mathrm{SCT}}}(t)$ is then used as the initial end-effector target pose for the next iteration. This process is illustrated in Fig. 4.10.

4.6 Target pose correction

If the assistance needs to be improved, the user can be given the ability to adjust it. Subsection 7.3.2 explores the adaptation of a learned SCT skill. Another potential cause of assistance failure is the perception system; for example, the target object's pose might be incorrect due to various factors such as unexpected lighting conditions, occlusions, an inaccurate object model, or camera decalibration. A typical scenario is pouring, where the tip of a grasped container is constrained to stay above a target object. If, for example, the target object's pose is off by 5 cm, the user cannot pour into it using the *pour* skill. To address this, a trigger allows the user to switch from an active instantiated SCT skill to a 'target pose correction' mode. In this mode, the end-effector is typically moved with translational control, as in the pouring example, but this can be task-specific. Any movement applied to the end-effector in correction mode is applied directly to the target instance. When the user exits this mode, the assistance resumes from where it left off, but with the updated pose. In the case of pouring, this adjustment aligns the target object in the world model with the actual object, potentially correcting any position offset. This method is effective because the assistance is object-centric and time-independent, allowing users to develop an understanding of the robot's behavior. Hence, a target misalignment of 5cm can be easily detected based

on the robot's actions and corrected by the user. Additionally, object instances in the world model are projected onto the camera image displayed in the user interface, providing visual confirmation of whether the pose estimate is accurate.

4.7 Conclusion

This section introduced Shared Control Templates, which feature a novel formalization for input mappings and active constraints, used in conjunction with a frame interpolator. Shared control skills are modeled as finite-state machines, effectively capturing the various phases of activities of daily living. The framework decouples assistance from user intent estimation, ensuring consistent behavior and controller independence, thereby enhancing user safety. The skills are object-centric, time-independent, and repeatable, providing transparent assistance. A wide range of behaviors can be developed to design robust skills and adapt to individual user preferences. However, a key limitation so far is the reliance on manually designed skills, which does not easily scale.

The following sections will show the results of user studies with able and motor-impaired users, followed by different methods to design skills from demonstrated robot trajectories.

Chapter 5

User experiments

Before describing how to semi-automatically learn SCTs in chapter 6 and 7, we will see in this chapter how we evaluated the interest of SCT in real applications.

Several studies were conducted during this thesis, primarily focusing on evaluating how effectively users could utilize the EDAN system to perform activities of daily living. Section 5.1 presents a pilot study involving participants without motor impairments. In section 5.2, results from a user study focused on motor-impaired users' ability to perform activities of daily living using EDAN are highlighted. Finally, section 5.3 covers the preparation and outcomes of the EDAN team participation in the Cybathlon Challenges 2023 and Cybathlon 2024 [Jae+23], where a motor-impaired pilot completed tasks of daily living using EDAN.

For each study, all participants gave written consent to the procedure, which was explained to them orally and in writing. The studies adhered to the guidelines outlined in the Declaration of Helsinki, and the ethics committee's approval was solicited when required.

5.1 Study with participants without motor impairments

The results in this section have been partially presented at ICRA 2020; see [Que+20]. In our first pilot study, three able-bodied users were asked to perform activities of daily living with the assistance of Shared Control Templates (SCT). These tests were conducted to illustrate the effect of our approach on the trajectories of the end-effector.

5.1.1 Study design

Three everyday life tasks were selected for evaluation: *Open drawer*, *Pour water* (see Fig. 4.1) and *Open door*. Three able-bodied participants performed these tasks using two continuous, 3 DoFs interfaces: a Spacemouse and an EMG-based interface. The participants had varying levels of experience with the system: P-A was an expert user of both the system and the interfaces, P-B had experience with the EMG-based interface but no prior knowledge of the system or method, and P-C had experience with neither. Each participant attempted to complete the tasks with shared control four times with the Spacemouse, then again with the EMG-based interface. For each task and user interface, the first three trials were used for training, with advice on the task execution being provided, while the final trial – performed without any external guidance from the experimenter – was used for evaluation. To conclude the experiment, participants also attempted the *Open drawer* and *Pour liquid* tasks using direct control with a 3 DoFs Spacemouse.

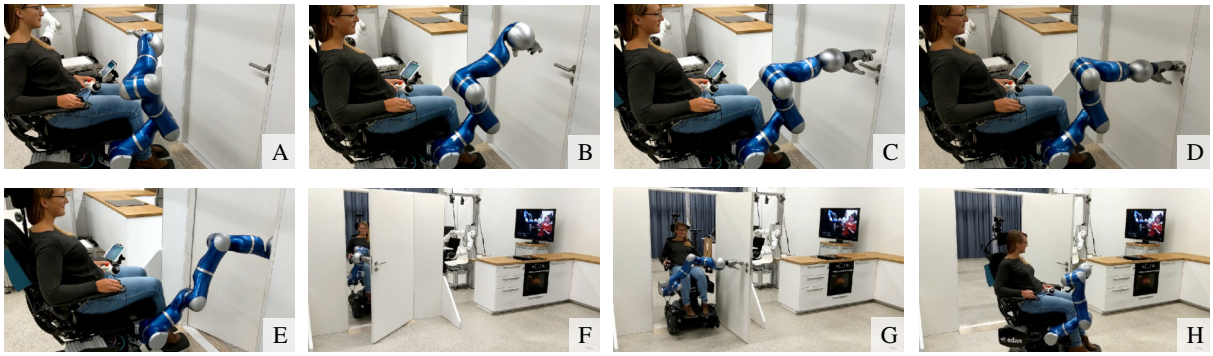


Figure 5.1: **Photo series of the different phases of the skill *Open door* with one of the participants. A**: approach and alignment to the door, **B-F**: open the door and **G-H**: drive through, executed using SCTs and whole-body control on the EDAN system with a Spacemouse.

5.1.2 Quantitative results

When using the Spacemouse interface, all three participants successfully completed the shared control tasks. When using the EMG-based interface, P-A completed all shared control tasks successfully (see Fig. 5.1 for one of the executions), P-B managed to open the drawer and go through the door multiple times but not pour water, while P-C was unable to complete any trials.

The time taken to complete the task *Pour water* during the test trials, along with expert user results, is presented in Table 5.1.

Task: *Pour water*	Participants (Evaluation (last run) average)	Skill designer (average of 10 trials)
Shared Control	37s	30s
Direct Control	95s	67s

Table 5.1: Time to completion of the task *Pour water* in shared control and direct control by the participants and the SCT designer.

Using SCTs consistently resulted in faster task completion than direct control for study participants and the skill designer. Participants also reported a preference for shared control. We observed that direct control was particularly time-consuming for less experienced users for two main reasons. First, orientation control in direct control is challenging since it is difficult to predict in which direction the end-effector will move from which command DoF, based on the current end-effector orientation. Second, direct control requires frequent mode switching, with an average of 6.7 switches in the expert trials. Even users familiar with the task need to alternate between translation and rotation control to properly align the tip for pouring, because rotational commands in direct control are applied to the tool center point of the end-effector rather than the tip of the grasped object. In contrast, in shared control of *Pour water*, the tip of the grasped object is used as the origin of the rotation axis.

In direct control mode, participants sometimes failed the *Open drawer* task due to exceeding the torque safety threshold (see subsection 3.4.3). This was subsequently improved by creating virtual walls to satisfy maximum allowable forces, as defined in subsection 3.4.4 and subsection 4.4.1. The *Open door* task was not attempted in direct control mode, as it is too complex to complete within a reasonable time frame – especially with EMG – due to the precise coordination required between the wheelchair and manipulator's DoFs.

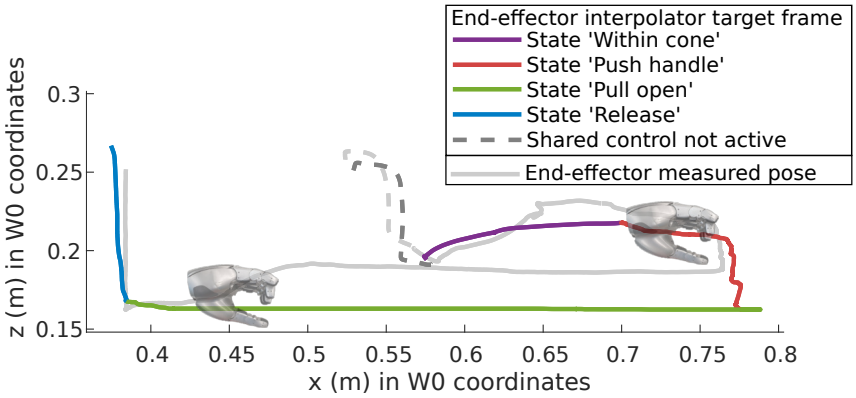


Figure 5.2: **Open drawer.** Side view of the trajectories resulting from opening the drawer by P-A during their evaluation trial with an EMG-based interface. The trajectories from the target SCT pose $\mathcal{E}_{\mathrm{SCT}}$ (in color) and the measured pose $\mathcal{E}_{\mathrm{MEASURED}}$ (in grey) are displayed. In the state 'Pull open', $\mathcal{E}_{\mathrm{MEASURED}}$ follows a parallel trajectory to $\mathcal{E}_{\mathrm{SCT}}$ due to the impedance control and the force applied to the drawer handle.

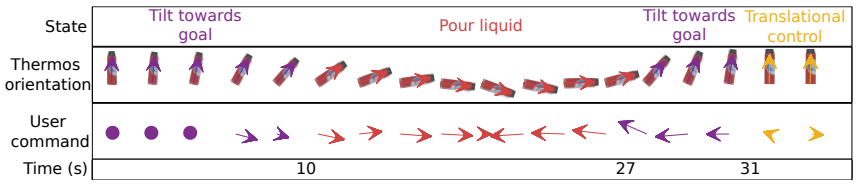


Figure 5.3: **Pour water.** Timeline of the task *Pour water* during P-A evaluation trial with an EMG-based interface.

5.1.3 Qualitative results

We present the results from P-A test trials with the EMG-based interface to illustrate the properties of our shared control method.

5.1.3.1 Open drawer

According to the users' feedback, the easiest of the three tasks was the drawer opening, shown in Fig. 5.2. The input mapping is translational control with state-dependent scaling. During the state 'Pull open', no downward motion is possible and lateral motions are scaled down, resulting in the observed smooth trajectory, even with a noisy EMG-based interface.

5.1.3.2 Pour water

A timeline of the task 'Pour water' is shown in Fig. 5.3. The state 'Tilt towards target' constrains the grasped object's tilt angle relative to the target's distance. Additionally, the end-effector is constrained to be perpendicular to the direction towards the target, ensuring that the pouring motion primarily uses the wrist joint. This adjustment increases the manipulator's workspace for this specific task. Once the thermos's tip is positioned above the mug, the 'Pour' state activates, mapping the user input to enable the rotation around the tip of the thermos. This assistance creates a smooth trajectory for the bottle's orientation, conveying the current task intent to the user (by moving the grasped object pointing towards the estimated target) while providing intuitive control.

5.1.3.3 Open door

As shown in the photo series in Fig. 5.1, the skill execution for *Open door* is detailed in Fig. 5.4, which highlights the smooth constraints applied to the end-effector. State 'Within cone' keeps the end-effector within a cone constraint, directed towards the door handle's grasp frame (cf Fig. 4.3, *Center*) as well as enforces an appropriate end-effector orientation for this task. During the 'Push door' state, the end-effector is constrained to follow a cylindrical trajectory, with a limit on the downward force acting in $-z$ (subsection 4.4.1). The input mapping and whole-body control allow the user to efficiently complete this complex task using primarily forward arm commands in x. When passing through the door in whole-body control – with only an 11cm margin – an absolute orientation controller keeps the wheelchair aligned perpendicular to the door, as described in [Isk+19]. At any point, the user could instead give backward commands (in x), getting the manipulator to close the door and then release the handle and the wheelchair to automatically move backward. The execution of those different tasks can be seen in the video accompanying [Que+20].

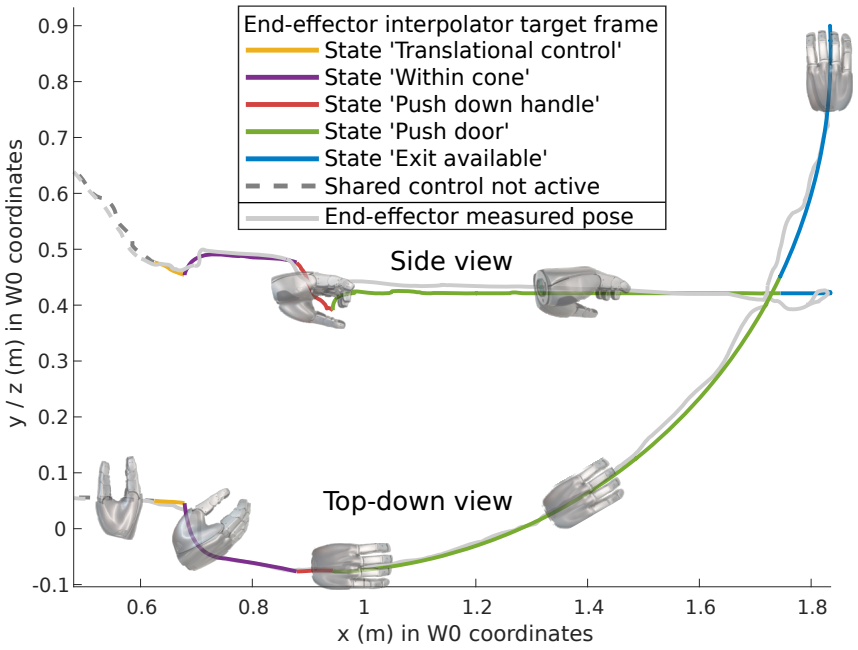


Figure 5.4: **Open door.** Top-down and side view of the trajectories resulting from $\mathcal{E}_{\mathrm{SCT}}$ (in color) and $\mathcal{E}_{\mathrm{MEASURED}}$ (in grey) during P-A evaluation trial with an EMG-based interface.

5.1.4 Discussion

These findings demonstrate that when using SCTs, users control key task actions, such as determining the amount of water to pour or how far to open a door. At the same time, the robot ensures that no water is spilled and that the end-effector correctly grasps the various objects. Regardless of the level of assistance, tasks were completed more quickly with shared control than with direct control. This efficiency stems from the simplified user control within a constrained workspace and eliminating the need for mode switches during task execution. One should note that time to task completion is but one metric to evaluate shared control; for example, although full autonomy might complete tasks faster, users may prioritize control over speed. Some aspects of the user intention, such as the quantity of water the user might wish to pour, are also challenging to estimate. Thus, the results suggest that SCTs enable the user to dictate the manner of task execution to meet their needs better than an autonomous approach could do.

The results also indicate that prior experience with the interface significantly impacts task success. They showcase that an EMG-based interface requires practice for effective use [HV18], and mastery of the interface helps to assess the impact of the assistance.

Implementing and testing our method in a realistic environment uncovered several challenges. For instance, some trials involving direct control failed when the safety torque threshold was exceeded. This occurred because, in direct control, the robot lacks awareness of the environment it is interacting with, making it impossible to adjust the safety torque threshold for object interactions: the system cannot distinguish between intentional contact and safety-critical situations.

The input mapping in the tasks *open drawer* and *open door* allows experienced users to adjust for target pose estimation errors. In contrast, the *pour water* skill implementation is more constrained. For the latter, the more complex input mapping creates an implicit workspace manifold (the poses available to the end-effector according to the input mappings and active constraints) that does not adequately cover the space of model errors (such as the bottle position in the horizontal plane). One possible option would be using a different input mapping, with u_1 and u_2 mapped to horizontal motion while u_3 is mapped to rotation around the tip of the grasped object. This does not rely as much on the object pose estimation, but as a result, it requires more precise user input and is less intuitive. In practice, the perception is usually correct, allowing us to guide users smoothly. If the assistance is wrong in the *pour water* task, the user has two options: either switch to direct control, or use a target pose correction mode, as described in section 4.6. The world model can then be adjusted to match the real environment, hence adjusting the behavior of the assistance.

5.2 Activities of daily living with participants with motor impairments

SCTs were also used in a user study conducted by Hagengruber *et al.* [Hag+25], where three motor-impaired users (P-D, P-E, and P-F) used EDAN.

The goal of this study was to evaluate an assistive device, EDAN, with the *actual target group*, in *realistic (home) environments* (not only in the laboratory, when possible), and on *long sequences of tasks* that naturally occur in everyday life (not only on isolated benchmark tests). Three participants did sequences of activities of daily living with SCT assistance using an EMG-based interface or a hybrid EMG and joystick interface. The study was conducted for P-D in our laboratory and for P-E and P-F in their own home. Seat settings and virtual workspace boundaries (subsection 3.4.2) were customized for each participant.

The study was approved by the ethics committee of the Technical University of Munich, School of Medicine (approval number: 6/14S). Additionally, all participants provided written consent for the publication of identifying information, including images and videos, for use in scientific publications.

5.2.1 Study with a motor-impaired participant in the DLR-RMC Re-enabling Robotics laboratory

The first participant, P-D, suffers from dystrophy Becker-Kiener (Type 43). Residual muscular activity allows him only limited upper limb movement, while tasks involving load or requiring outstretched arms, such as drinking from a bottle or opening a door, are not possible. P-D uses a commercial 2D joystick in daily life to move his wheelchair. In this study, he used a hybrid interface, with two DoFs controlled with a Spacemouse and one DoF with an EMG-based interface. He had no prior experience with the EMG-based interface or the EDAN system.

Making use of the full functionality of the EDAN system, P-D was successfully able to do a sequence of tasks using different control modes: direct control (DC), direct control with whole-body control (DC-WBC), shared control (SC), and shared control with whole-body control (SC-WBC):

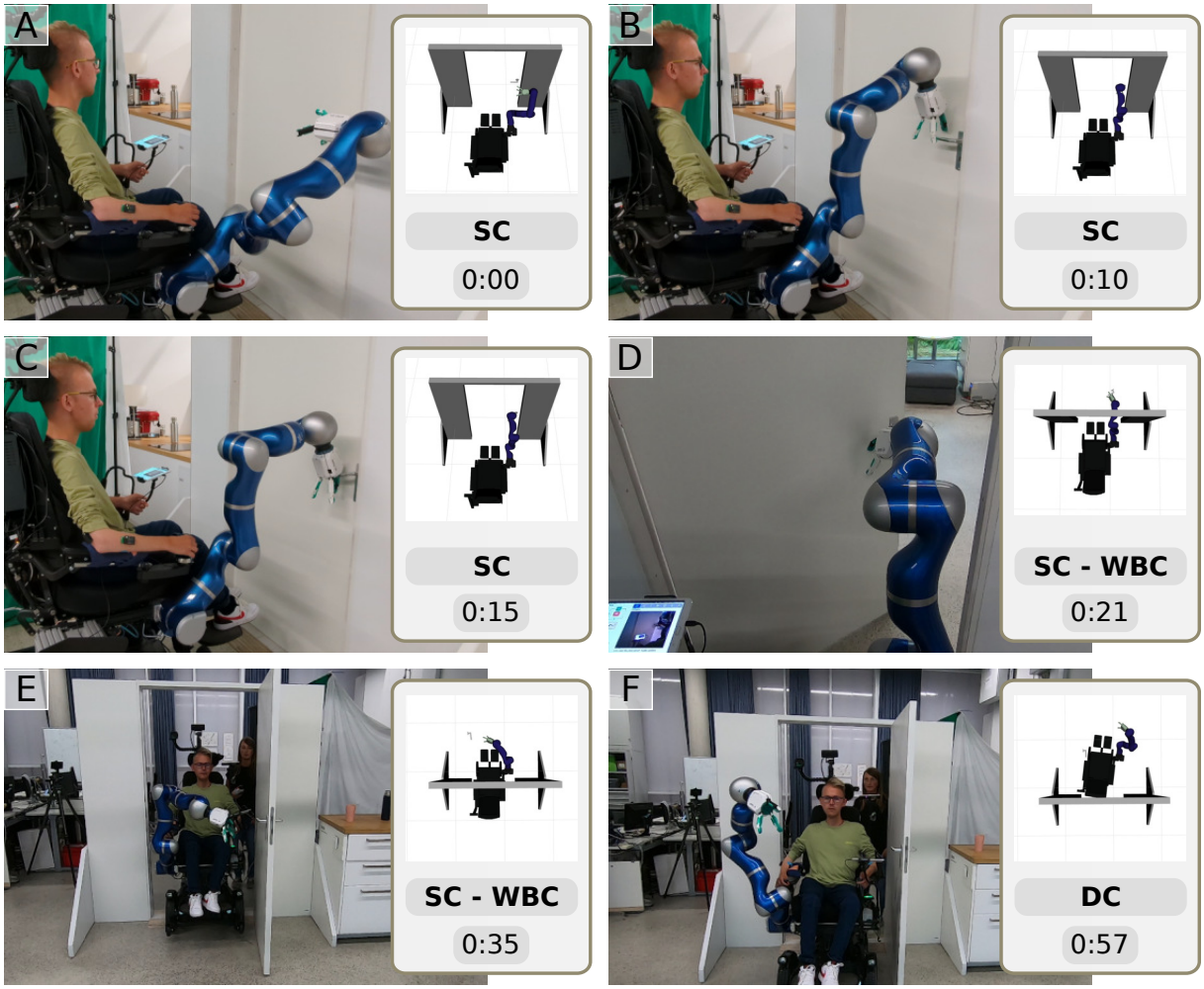


Figure 5.5: **Photo series of opening and driving through a door**. The figure shows P-D using the hybrid interface to perform the *open door* task with shared control with whole-body control (SC-WBC); the robot world model, the control scheme, and the times are visualized. **A**: Starting the shared control task, with the robotic arm next to the handle. **B**: The position and orientation of the arm are guided by the SCT to place the robotic hand above the door handle. **C**: Pressing the door handle. **D**: Opening the door in a circular motion while the wheelchair follows through the door. **E**: Releasing the door handle. **F**: Driving the remaining path through the door, using direct control. Adapted from [Hag+25]

1	Open then go through a door (see Figure 5.5)	SC-WBC
2	Open a drawer	SC-WBC
3	Pick mug from the drawer	DC
4	Close the drawer	DC
5	Place the mug on top of a kitchen counter	DC
5	Open a fridge	SC-WBC
7	Pick a bottle	SC
5	Close the fridge (with bottle grasped)	DC-WBC
5	Drive back to kitchen counter	DC
8	Pour liquid into mug	SC
9	Place the bottle onto kitchen counter	SC
10	Pick the mug	SC
11	Drink from the mug	SC
12	Place the mug onto kitchen counter	SC

5.2.2 Study with participants with motor impairments in their own home

Both participants suffer from spinal muscular atrophy type II. This disease leads to the death of the motor neurons in the spinal cord, resulting in progressive muscular atrophy, and both were dependent on 24-hour care. Despite its strong progression, both could evoke voluntary muscle activation at different locations along their arms, which were measurable with an EMG-based interface.

The participants used the robotic system in their homes in 6 and 5 experimental sessions, respectively. They had to be familiarized with the interface and the core functionalities provided by the system. Therefore, the complexity of the tasks performed by all participants increased from session to session.

The aim was to perform a sequence of activities of daily living in the last session:

1	Open a drawer	SC-WBC
2	Pick mug from the drawer	DC
3	Close the drawer	DC
4	Place the mug on top of the cabinet	DC
5	Move the wheelchair closer to the cabinet	DC
6	Pick a bottle	SC
7	Pour liquid into mug	SC
8	Place the bottle	SC
9	Pick the mug	SC
10	Drink from the mug	SC
11	Place the mug on the table	SC

To get acquainted with the EDAN system, P-E and P-F first performed the subset of tasks 6-11 multiple times with shared control. In the fourth session, they performed this subset in 4:15 min and 3:39 min, respectively. Then, in her sixth session, P-E performed the entire sequence while sitting in EDAN in 9:18 min, as illustrated in Fig. 5.6 [1].

5.2.3 Discussion

This study showed that users with motor impairment were able to execute a sequence of activities of daily living with the EDAN system assisted by SCTs with whole-body control. See [Hag+25] for details on the study setup, the results, and their detailed discussion. The discussion on evaluating shared control systems

[1] Due to a lack of qualified caregivers, P-F could only sit in Edan in a single session, controlling the system remotely from her bed in the others. For the same reason, session 6 could not be realized for P-F.

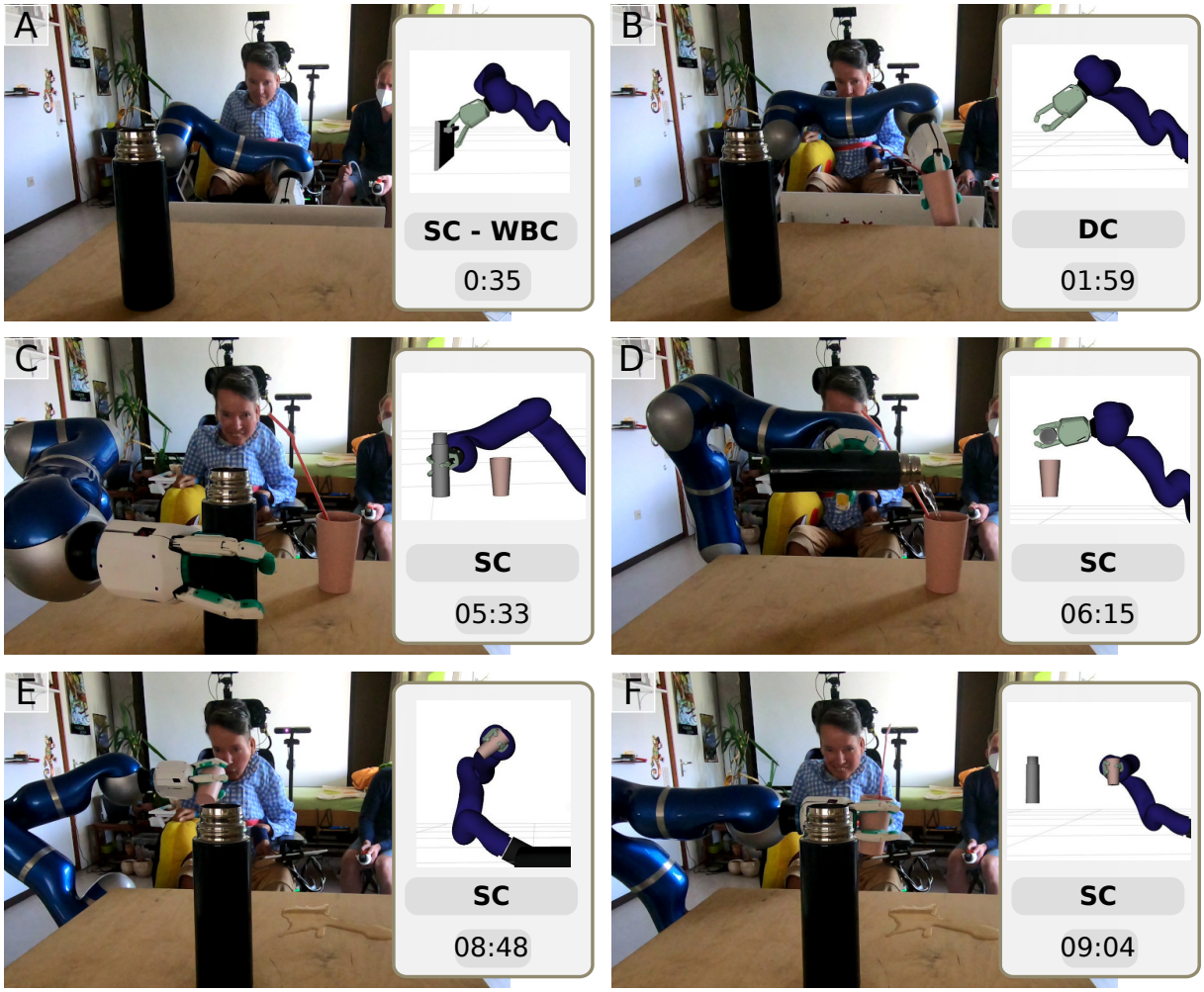


Figure 5.6: **Photo series of the successively executed sequence of tasks using different control schemes.** P-E is shown seated in EDAN while performing the tasks. The time sequence started from the default robot position. **A**: Open the drawer using SC-WBC. **B**: Pick a mug from the drawer using DC. **C**: Pick a bottle with SC. **D**: Pour into the mug with SC. **E**: Drink from the mug using SC. **F**: Release the mug again with SC. Adapted from [Hag+25].

and analyzing the strengths and limitations of SCTs can be read in section 5.4.

5.3 CYBATHLON

CYBATHLON, a non-profit initiative by ETH Zurich, serves as a platform that challenges teams worldwide to develop assistive technologies suitable for everyday use with and for people with disabilities [Zür]. At the heart of CYBATHLON are international competitions and events where teams comprising technology developers from universities, companies, or NGOs partner with individuals with disabilities to complete everyday tasks using their latest assistive technologies. Participants demonstrate tasks like tying shoelaces using a robotic arm prosthesis, balancing on uneven surfaces with a prosthetic leg, or navigating rugged terrain with an exoskeleton. Beyond the competition itself, CYBATHLON provides a benchmarking platform to advance research on assistive systems for everyday challenges while fostering public dialogue to promote the inclusion of people with disabilities in society. The pilot's active participation is seen as

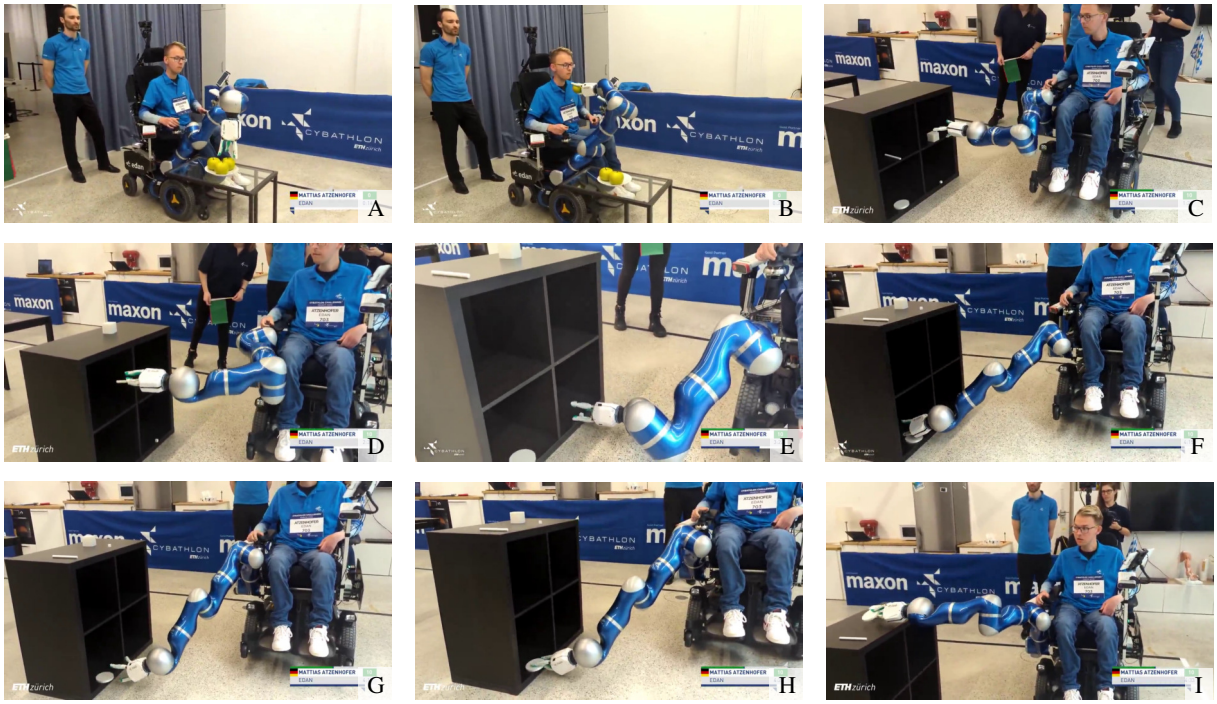


Figure 5.7: **2023 CYBATHLON Challenges.** The pilot carried out the two tasks within 5 minutes: picking and biting an apple and taking objects from a shelf.

crucial in both the competition and development process to ensure that the needs and perspectives of end users are considered and addressed.

The CYBATHLON takes place every four years, with the third edition scheduled for October 2024. It features a race where pilots with motor impairments must complete various tasks. Several race categories are available for the 2024 event: arm prosthesis, vision assistance, brain-computer interface, exoskeleton, leg prosthesis, wheelchair, functional electrical stimulation bike, and assistive robot. CYBATHLON Challenges are held between the main events, showcasing a subset of tasks of the main event. The assistive robot race, which was recently added, involves motor-impaired pilots performing activities of daily living with an assistive robot. We demonstrated our technology by winning the 2023 CYBATHLON Challenges and the 2024 CYBATHLON.

5.3.1 2023 CYBATHLON Challenges

The 2023 CYBATHLON Challenges featured two tasks to be completed within a 5-minute time limit. The first task involved grasping one of three apples from a plate, simulating a bite by bringing the apple to the pilot's mouth and returning it to the plate. The second task required grasping four objects from randomized positions in a shelf: a die, a cube, a tube, and a disk. The pick-up order was revealed during the event. After each object was placed on top of the shelf, the judge indicated the next object. The pilot had to drive around the table where the next object was indicated to determine the next object.

SCTs supported the pilot for both tasks. For the *pick apple* task, he could select which apple to pick with translational control, then bring it to his mouth, with the end-effector orientation based on its position relative to the wheelchair. The end-effector was guided to an object-specific grasp pose to pick objects from the shelf, and once picked, the placement on top of the shelf was automated. The disk was the most challenging to retrieve because it had to be slid along the shelf and then picked up. Due to its size with respect to the end-effector maximum opening, no direct grasp was possible. The pilot could initiate a

regrasp in case of a failed grasp: he could trigger the opening of the end-effector fingers, reposition the end-effector, and trigger again to grasp.

The pilot participated in six 3-hour practice sessions at our laboratory to prepare for the event. During the first session, the wheelchair was adjusted, and virtual boundaries were set for safety, helping him build confidence in the system. He then practiced picking up the various objects before completing entire trial runs. SCTs were iteratively refined during those practice sessions based on the pilot's performance and his feedback on the assistive behavior.

On the day of the 2023 CYBATHLON Challenges, the pilot completed two runs, successfully finishing both tasks within the required time limits and winning the challenge, with times of 4:44 and 4:28, respectively. The event's video recording excerpts are shown in Fig. 5.7. A video can be seen at `www.youtube.com/watch?v=EoER_5vYZsU`

5.3.2 2024 CYBATHLON

The 2024 CYBATHLON features ten tasks to be completed within a 10-minute time limit, with points accorded for each completed task; see also [Zür]. The tasks are:

- *Mailbox*: Open a mailbox, extract a parcel, and transport it to a target location.
- *Toothbrush*: Pick and bring a toothbrush to one's mouth.
- *Pick up*: Pick up a bottle from the floor and place it on a table.
- *Scarf*: Hang a scarf on a clothesline.
- *Eating*: Pick and bring an apple to one's mouth.
- *Crowd*: Pass between individual furniture pieces while avoiding robots roaming the task space.
- *Spice up*: Pick up two objects (indicated by the judge) from a shelf and set them on a predefined location.
- *Door*: Pull open, pass through, and close a door.
- *Touchscreen*: Navigate a touchscreen and order a predefined food item.
- *Dishwasher*: Open a dishwasher and take out a plate.

The pilot came to our lab for a 3h session once a week for three months. Although originally we provided assistance with strong guidance, over time and from our pilot's feedback, we adjusted the skills to let the pilot be more in control of the motions, increasing his agency and the robustness of the execution. Excerpts from our pilot winning the competition can be seen in Fig. 5.8.

5.3.3 Discussion

The CYBATHLON is an interesting alternative to user experiments or simulations for evaluating assistive devices. Studies involving user experiments typically focus on isolated features or tasks, often conducted in controlled laboratory environments (though notable exceptions exist, particularly the work with Henry Evans [Cio+12; Che+13; Par+20]). However, full-fledged systems that handle sequences of everyday tasks, such as pouring oneself a drink before drinking, are seldom explored.

Another approach is simulated benchmarks for assistive robots. This approach is relatively new and faces common challenges associated with robotic simulations, compounded by the difficulty of accurately modeling human behavior, as discussed in section 2.9.

The CYBATHLON involves a single pilot preparing and performing activities of daily living under stressful conditions (with many spectators cheering), outside the laboratory, and within strict time limits. A key goal of the event is to ensure the pilot's active involvement in the development process, incorporating their perspectives and needs into the system's design. This had practical implications, such as the pilot choosing the DLR-CLASH hand over a rigid off-the-shelf gripper because he felt safer using it near his face, for example, while 'biting' into an apple.

Figure 5.8: **2024 CYBATHLON winning run A**: Mailbox. **B**: Toothbrush. **C**: Pick up. **D**: Scarf. **E**: Eating. **F**: Crowd. **G**: Spice up. **H**: Door. **I**: Dishwasher. Source: 2024 CYBATHLON.

When designing skills for the CYBATHLON, a balance has to be found between robustness, speed (which is crucial in this competition but not so much in daily use), user preferences, and agency. Since tasks are known beforehand, following along a prerecorded trajectory may sometimes be the fastest approach. However, it may not be the most robust for everyday use or preferred by the user. Take, for example, the *Mailbox* task. Even by only providing assistance with the SCT framework, various designs – hence behaviors – are available. The most straightforward approach involves two states with translational control, scaled input mapping, force limits, and a binary trigger to switch between the states to open and close the end-effector. This approach does not assist the user much and relies on fine motor control and a good view of the object. A second option with high assistance is to break the task into a series of subtasks, which can compensate for imprecise perception:

- Set fingers to a hatch grasp configuration and guide the end-effector in the correct orientation towards the mailbox using cone constraints.
- Scale down commands to slowly press onto the front of the mailbox.
- Move vertically upwards until a high enough force estimate indicates that the fingers are lodged in the hatch.
- Pull the hatch open.
- Once far enough from the mailbox, release the fingers to let the hatch drop open.
- Guide the end-effector above the parcel.
- Press down on the parcel.
- Slide the parcel out slightly to make it easier to grasp.
- Raise the end-effector slightly and open the fingers.
- Lower the end-effector.
- Close the fingers to grasp the parcel.
- Map one input DoF to a displacement in 3 dimensions: -$\boldsymbol{x}$, $\boldsymbol{z}$, and θ_y to follow a curved trajectory and smoothly extract the parcel.

A third and currently preferred option is to extract the parcel using scaled translational control, then grasp it upon user trigger, and finally follow a curved trajectory using the same input mapping as the last stage of the second option. This minimizes parcel movement and reduces the risk of dropping it. Additionally, if the grasp fails, the user can reopen the fingers, reposition the end-effector, and attempt to regrasp, as is the case with other skills mentioned before.

Those design examples highlight the flexibility of behaviors that can be achieved with Shared Control Templates and the trade-offs involved. For instance, increasing the number of pathways through the finite-state machine may improve robustness, but it can also make the system less intuitive for the user or more reliant on perception.

The performances obtained at the CYBATHLON Challenges and the CYBATHLON suggest that the SCT framework is well-suited for sequences of everyday tasks. It allows for the design of skills that range from highly constrained to low guidance with high user's agency. This adaptability is crucial, as aligning with user preferences is essential, as demonstrated by various studies [CTA21; Bha+20]. However, the downside of the current design is that this flexibility is an explicit design choice rather than a parameter that the user can easily adjust themselves, and designing SCTs is labor intensive. Chapters 6 and 7 investigate methods to mitigate this downside by facilitating the design of SCTs.

The CYBATHLON represents a significant step towards the real-world deployment of assistive technologies. However, this first edition of the assistive robots race had some limitations. First, there was a lack of diversity in the tasks. Critical activities like eating a meal and drinking, requiring skills such as scooping or pouring, were not considered. Second, the run itself lacked variability. The 2023 CYBATHLON Challenges task involved picking objects from a shelf. Different objects requiring different grasp strategies were randomly placed in the shelf and had to be picked in a specific order. In contrast, in

the 2024 edition, the objects of the *Spice up* task are similar and can be picked with the same assistive behavior. Those limitations, using known objects and a deterministic environment combined with the time constraint, encourage highly fine-tuned solutions, which conflict with the goal of "driving research on assistance systems forward for dealing with daily-life challenges". Introducing more variability to reduce the reality gap would be an exciting development, such as randomizing the order of tasks or adding tasks requiring usage of objects, such as scooping food and pouring.

5.4 Conclusion

Those results allow us to draw conclusions about evaluating shared control systems and the advantages and limitations of the Shared Control Templates framework.

5.4.1 Evaluating shared control systems

The evaluation of robotic systems is a well-known and difficult challenge due to the wide variety of robots, tasks, and environments encountered [Beh06]. For multiple reasons, these challenges are further complicated with a human in the loop.

Firstly, the choice of interface and the user's proficiency with it plays a crucial role, which affects the comparability of results achieved by different users negatively, as highlighted in Section 5.1. As discussed in subsection 2.1.2, the wide variety of motor impairments leads to a range of interfaces, each offering different DoFs and varying levels of ease of use depending on the technology involved. Additionally, every user is unique, with differing levels of motor control and experience with specific interfaces. Meaningful evaluation requires users to be assessed with an interface they are familiar enough with. For instance, a user's familiarity with an EMG-based interface might require at least several hours of use and can be assessed using tasks such as the Box and Blocks Test or the ARAT [HV18].

Secondly, there is the question of the evaluation criteria. When comparing two methods, the aim is not necessarily to determine which one is faster to get used to or allows for faster task completion. What matters more is typically which method would the users use in their daily life after they have become accustomed to both methods. These preferences can be influenced by different evaluation criteria, which may vary in prominence depending on the user. Some criteria are objective, such as reliability (e.g., not knocking over a glass when grasping it) or task completion time. Others are subjective, including ease of use, transparency of the robot's behavior, and the user's sense of control and agency. Those metrics are often evaluated with a Likert scale questionnaire. The works cited in section 2.8 provide many different evaluation metrics, depending on each study's respective goal.

Moreover, user preferences can evolve over time. Our motor-impaired participants reported prioritizing being in control over completing tasks quickly. If the robot behaves unpredictably or is not deemed valuable enough in shared control, users may prefer direct control, even if it takes more time and effort to complete a task. Long-term studies with devices in home environments are essential to better understand these preferences, with the results in subsection 5.2.2 a step in this direction.

Lastly, in shared control, the robot's behavior is shaped by the human's actions, particularly the commands given. Still, it can also include other cues such as gaze or opening one's mouth to receive a spoonful of food. Simultaneously, users adapt their behavior to the robot's actions to fulfill their goals, resulting in co-adaptation and complexifying the analysis. This effect is magnified when assistance systems consider the history of human actions, such as command history during a task, as done by Javdani *et al.* [Jav+18]. SCT's assistance relies solely on the current state and does not factor in past actions, thereby limiting this co-adaptive effect.

5.4.2 Shared Control Templates analysis

Shared Control Templates have proven effective in assisting users with motor impairments in performing task sequences in their homes and our laboratory, outperforming direct control. In direct control, users frequently need to switch between controlling different DoFs of the system, which becomes especially challenging for tasks requiring a wide range of motion and whole-body coordination, like opening doors or drawers. In such cases, users may face difficulties such as reaching singularities when fully extending the manipulator. Integrating shared control with whole-body control significantly helped address these challenges.

Including a user in the control loop offers a significant advantage for assistive robots. First, it enhances the user's sense of agency over their environment. Additionally, the system's reliability improves thanks to the user's situational awareness. Even if the robot assistance functions correctly only 95% of the time, the user can take over for the remaining 5%, or modify the assistance with target correction mode (section 4.6) or model adaptation mode (subsection 7.3.2). These two factors could contribute to the practical adoption of such robots.

According to cognitive psychology theories, a mental model represents how users understand a system. Explanations help users form a mental model of how the robot, particularly the assistance, behaves [MZR21]. SCTs rely solely on the robot's current state and the finite-state machine's active state. The user intent estimation and the active state can be visualized on the GUI. The skill constraints can also be displayed on the tablet with RViz [Kam+15] for informational purposes, but they are not intuitive enough to use during task execution. Those features make SCT an explicit representation of an assistance with predictable behavior. The separation of goal selection from the assistance further promotes transparency and helps users form a clear mental model of the assistance. This predictability is particularly advantageous for noisy interfaces, such as EMG-based interfaces. For example, suppose the robot does not move even though the user wishes it to. In that case, they might be tempted to give stronger signals to generate commands, which end up counterproductive, with signals outside the Gaussian process training distribution and hence null. Therefore, clearly understanding how the commands generate robot motion is crucial for smooth control, which SCTs support with their transparent behavior.

Encoding temporal information is avoided, as it forces the user to follow a pre-determined temporal behavior. This approach suits tasks like manipulating grasped or articulated objects, where the finite-state machine accurately matches the different phases of a task with transitions based on poses, forces, or user triggers. However, other tasks may benefit from incorporating temporal information, such as assistive dressing [PC17].

Ideally, assistive skills should provide users with maximum agency while ensuring the task cannot fail. In reality, assistance is imperfect due to the complexity and variability of the real world. SCT allows for adjustment in the degree of assistance, striking a balance between user agency and task robustness. Granting greater freedom in task execution might reduce robustness while restricting the user to a single trajectory might enhance reliability but could lead to frustration. Achieving maximum reliability in complex or challenging tasks often involves carefully balancing user agency and task space restrictions.

SCT is not meant for dynamic environments, although simple tasks like handovers can be handled. Generally, commanded motions are relatively slow compared to the robotic arm's technical capabilities and sufficient for most activities of daily living. Avoiding damage to the robot and unwanted effects on the environment is a high priority, second only to user safety. Efficiency in completing tasks is a lower priority, according to user feedback. Even in competitive settings like the CYBATHLON, where time is critical, we still operate relatively slowly. While greater autonomy could potentially lead to faster task completion, the pilot prefers maintaining control, especially regarding motion speed.

It is crucial to gather user feedback on what they would like the robot to do. For instance, a robot could assist in adjusting bedsheets at night to regulate temperature, eliminating the need to call for help. That's one of the benefits of events like the CYBATHLON: with repeated sessions and continuous explanations,

even non-technical users gain a better understanding of what the robot can do, cannot do, and might be able to do with further research. In return, users can give more informative feedback on what they would use such an assistive device for. Users who already have wheelchair-mounted arms with direct control could also provide feedback.

5.4.3 Future work

Obtaining more qualitative and quantitative results would be beneficial, as SCT has not yet undergone testing with a large group of users. Additional tasks, particularly those identified as important by users – such as eating [Bha+20] – should be evaluated. SCT could initially be compared to a baseline, for instance, by evaluating users with an assistive device that lacks shared control and measuring task performance and user satisfaction for direct control and shared control with SCT. Ideally, various shared control approaches should also be compared, though this would require users to become proficient in multiple methods, necessitating significant practice time.

The following two chapters consider the problem of learning skills from demonstrations to facilitate skill design.

Chapter 6

Learning parameterized SCT active constraints from human demonstrations

The results in this chapter have been partially presented at IROS 2021, see [Que+21].

6.1 Introduction

As previously detailed, SCTs assist with activities of daily living, using components such as input mappings and active constraints. However, a shortcoming of SCTs as described so far is the reliance on an expert programmer to manually code a finite-state machine with input mappings, active constraints, and robot parameters for each task. To address this issue, we investigated learning SCTs from demonstrations, aiming to facilitate the development of new shared control skills. This chapter introduces a method for semi-automatically learning active constraints for SCTs from demonstrations with parameterized constraints fitting, as illustrated in Fig. 6.1. A set of end-effector trajectories is first recorded by kinesthetic teaching and then segmented based on trajectory curvature or contact forces. Constraints can subsequently be learned from these segments. The proposed algorithm in section 6.3 aids in determining which models best represent the constraints underlying the various phases of a task.

Additionally, demonstrations for a new SCT can be compared to a library of existing SCTs to identify states matching the demonstrations. This allows the transfer of components from existing SCTs to the new one, achieved using two comparison metrics designed to identify known states that match the new demonstrations, detailed in subsection 6.3.5. This approach can reduce the number of demonstrations required to construct an SCT or the number of parameters that need to be manually specified.

To validate our approach, we verified that the SCTs generated from the demonstrations effectively support users in performing complex tasks of daily living. This evaluation was done on the EDAN robot and described in section 6.4.

6.2 Constraint representation

Certain activities of daily living involve highly constrained motions, such as pulling open a drawer, which can be modeled using a prismatic constraint. In contrast, actions like approaching the drawer handle can be guided by a cone constraint (see Fig. 6.2), providing the user more freedom in moving the end-effector and allowing the option to switch tasks by moving away. Extensive research on constraint representations exists [BDB13], from which we employ two types of models: parameterized surfaces and volumes in 3D Euclidean space.

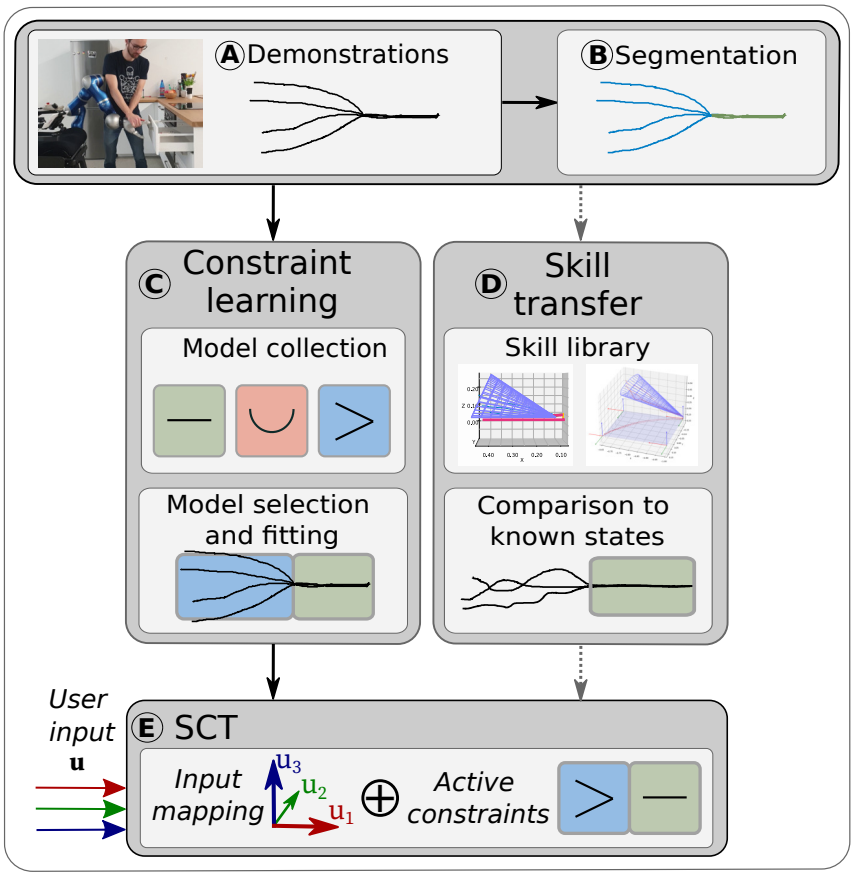


Figure 6.1: **SCT design method. A.** Gathering of demonstrations via kinesthetic teaching or direct teleoperation. **B.** Data segmentation according to contacts or trajectory curvature. **C.** Models are selected and fitted to represent the constraints of each task phase. **D.** Optionally, information from previously learned SCTs can be used if phases are similar. **E.** An SCT is represented as a finite-state machine with state-specific input mappings and active constraints using the previously learned constraints.

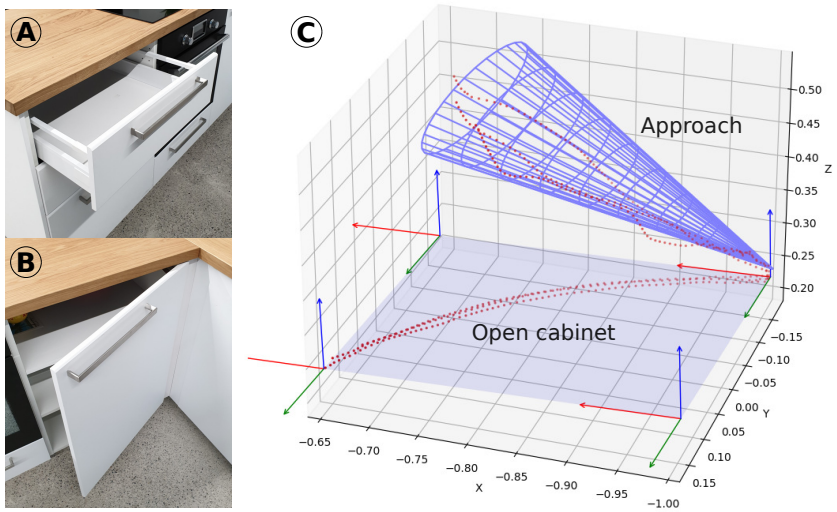


Figure 6.2: **Examples of constraints. A.** A prismatic motion is needed to open the drawer. **B.** An axial rotation is needed to open the cabinet. **C.** Two phases from the data recorded by opening the cabinet door, fitted with a cone and a plane, respectively. Those constraints may not be the optimal fit to build an *open cabinet* skill and are displayed here as examples.

6.2.1 Parameterized surfaces

Explicit constraints, such as lines or axial rotations, are essential for manipulating constrained objects, like pulling a drawer or opening a door. These constraints ensure consistent behavior from the user's perspective and can be associated with specific semantic meanings, such as whether a drawer is open or closed. The constraints utilized in this work, along with their respective DoFs, split into translation and rotation, are: plane (2 translational, 3 rotational), planar (2,1), line (1,3), prismatic (1,0), arc circle (3,0), and axial rotation (0,1).

6.2.2 Parameterized volumes

A parametric representation of explicit volumes, such as cones, curved funnels, or cylinders, can offer practical and easily recognizable constraints. These volumes are valuable for guiding user motion while allowing for some flexibility in fine control, for instance, at the beginning or end of a task. Additionally, they can be used to restrict the end-effector to safe regions.

The projection function on those volumes can be on the surface, within the volume or outside the volume, see subsection 4.3.2. The projection type can be specified when designing the skill and depends on the desired behavior.

6.3 Interactive design procedure of parametric constraints from human demonstrations

The pipeline for learning and designing a new SCT is depicted in Fig. 6.3 and presented in this section.

6.3.1 Data acquisition through robot demonstrations

To build an SCT, multiple demonstrations of the whole task are first recorded to capture variability within the task. Typically, five to ten demonstrations are sufficient for the considered constraints. The recorded

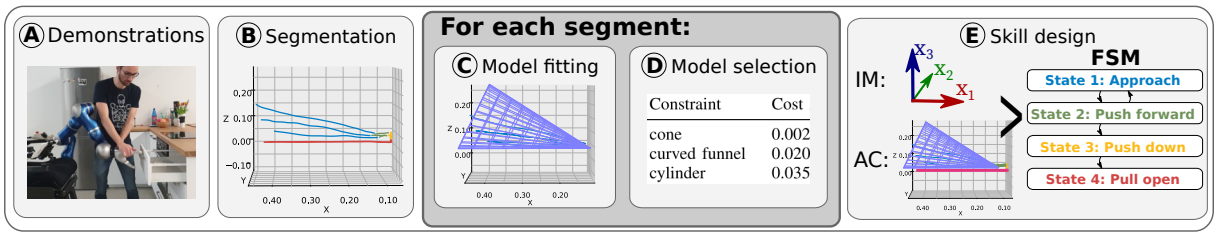


Figure 6.3: **Pipeline for interactive SCT design. A.** Demonstration data is acquired. **B.** Recorded data is segmented. **C.** Multiple constraint models are fitted for each segment. **D.** The constraints are heuristically ordered based on a cost to help the SCT designer select the most appropriate. **E.** Finally, the input mappings and the active constraints of each state are specified by the SCT designer and assembled as a finite-state machine.

data, composed of end-effector trajectories, can be acquired from teleoperation or kinesthetic teaching. An object-centric SCT representation is achieved using the target object as the reference coordinate.

6.3.2 Segmentation

The recorded data is segmented to create a set of segments S, with each segment containing portions of each trajectory. This segmentation groups data from trajectories influenced by the same constraints, thus defining different states of an SCT and enabling the modeling of active constraints.

Trajectories are first pre-processed with Dynamical Time Warping [SC07] to synchronize timestamps. Segmentation candidate points are then identified in each demonstration, using either sharp turns in the curvature [Cal09] or contact forces at rising and falling edges while adhering to a minimum segment length requirement. A segmentation point is retained only if corresponding points are present in all other trajectories within a close time window.

For instance, the segmentation process yields four segments for the demonstrations of the task "opening a drawer," as illustrated in Fig. 6.3.B. The first segmentation point arises when the end-effector reaches a pre-grasp position, the second when it makes contact with the drawer, and the third when it begins to pull the drawer open. The contact force and its location are detected and estimated using a momentum-based framework [Isk+21].

6.3.3 Constraints fitting

Once the data is segmented, geometric constraints can be assigned to each segment (see Fig. 6.3.C & 6.3.D). For every segment, the geometric models outlined in subsection 6.2.1 and 6.2.2 are fitted on the trajectories. The parameters of the parameterized surfaces are learned by fitting these geometric models to the recorded data, as described in [SZG18]. For the parameterized volumes, models are first initialized on a few random points. For example, three points are enough to define a plane or a cone constraint. The constraint is then optimized with the CMA-ES algorithm [HMK03], which minimizes the cost function defined below. The best result from M runs is selected for each model to minimize the risk of converging to a poor local optimum.

A cost value is determined for each class of constraint models, assisting the skill designer in selecting the most suitable constraints. The cost functions make use of a custom distance metric $Dist$ between two poses $\boldsymbol{H}_A$ and $\boldsymbol{H}_B$, defined as:

$$Dist(\boldsymbol{H}_A, \boldsymbol{H}_B) = \|\boldsymbol{p}_A - \boldsymbol{p}_B\| + \alpha \cdot |\theta|, \tag{6.1}$$

with the Euclidean distance in meters, the rotation angle θ of $\boldsymbol{R}_A {\boldsymbol{R}_B}^{-1}$ in an axis-angle representation in radian, and α a weighting term.

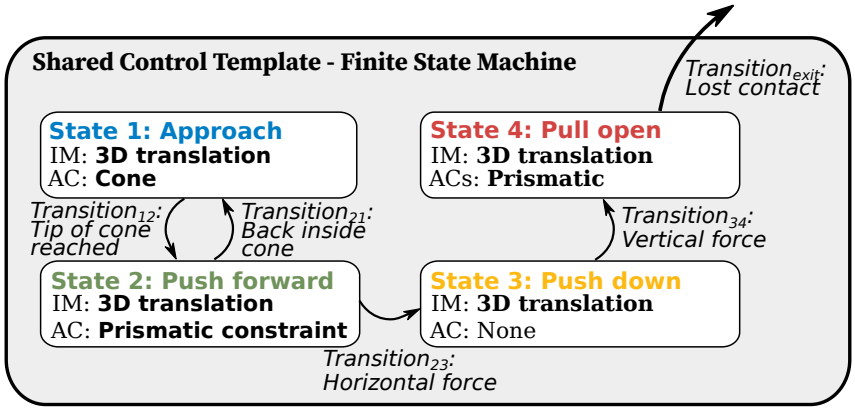


Figure 6.4: **Results of the pipeline from Fig. 6.3.** We show an SCT - in the form of a finite-state machine - semi-automatically built from demonstrated data. The SCT consists of four states with input mappings (IM) and various active constraints (AC). The transitions were specified by the skill designer.

The cost function for a parameterized surface model m_S is defined using the project function defined in Equation 4.4 and Equation 6.1 as

$$\text{cost}_{\text{surface}}(m_S) = \frac{1}{N}\sum_{n=1}^{N} Dist(\boldsymbol{H}_{\mathcal{E}}(n), \text{project}_{m_S}(\boldsymbol{H}_{\mathcal{E}}(n))) \tag{6.2}$$

with number of recorded poses N and recorded pose $\boldsymbol{H}_{\mathcal{E}}$.

The cost function for a parameterized volume model m_V is also defined using (4.4) and (6.1) as

$$\begin{aligned}\text{cost}_{\text{volume}}(m_V) = &\frac{1}{N}\sum_{n=1}^{N} Dist(\boldsymbol{H}_{\mathcal{E}}(n), \text{project}_{m_V}(\boldsymbol{H}_{\mathcal{E}}(n))) \\ &+ \beta \cdot \text{volume}(m_V)\end{aligned} \tag{6.3}$$

with β a regularization factor, leading to higher costs for bigger volumes (otherwise, an infinite volume would have a null cost with $\beta = 0$).

After calculating the costs for each geometric constraint, the models are ranked by their costs within each model class (surfaces and volumes). An example is shown in the results section in Table 6.1. Models from different classes are not compared with each other, as their costs are derived from distinct cost functions. These cost rankings, along with visualizations of each model, help the skill designer select the most appropriate geometric constraint.

6.3.4 Finite-state machine design

Once active constraints have been learned for each state, the skill designer can finalize the skill representation. First, transitions between states must be implemented to build the finite-state machine. If the default translational mapping is not appropriate, the input mapping of each state can then be specified, for example, by scaling down motions in DoFs perpendicular to the direction of motion. Finally, robot parameters can be configured (see subsection 4.4.1 for details). This process is illustrated in Fig. 6.3.E.

For the *open drawer* example, the outcome is shown in Fig. 6.4. In this case, the skill designer defined the transitions as the distance between the end-effector and the drawer handle (transition between states 1 and 2) and a force threshold (for the others). A translational control with scaled input mapping was selected for each state, leaving the user unable to control the orientation, which is unnecessary for this task.

6.3.5 Transferring knowledge for new shared control skills

Our multi-model finite-state machine representation can optionally be used to bootstrap the learning of new SCTs from a library of existing SCTs, for example, to reduce the number of required demonstrations or transfer hand-defined parameters. As a concrete example, after the SCT *open drawer* has been learned from demonstrations, it could help build a new SCT *open cabinet door*.

Although the constraints for the final phase of opening a cabinet door differ, the constraints for the initial three states, which guide the grasping of the handle, can be transferred from the *open drawer* SCT. This is possible because the drawer and cabinet share the same type of handle in our experimental kitchen setup, as illustrated in Fig. 6.2.

Given a set S of segments from newly demonstrated and segmented data, each segment s is compared to every state q of a learned SCT of interest with two similarity metrics. The first metric, L_{AC}, only takes the active constraints into account and considers the demonstrated data as a set of poses. A set of N_s poses consists of all poses of all segments of trajectories of s. L_{AC} is defined similarly to Equation 6.2:

$$L_{AC}(s,q) = \frac{1}{N_s}\sum_{n=1}^{N_s} Dist(\boldsymbol{H}_{\mathcal{E}}(n), \text{project}_q(\boldsymbol{H}_{\mathcal{E}}(n))) \tag{6.4}$$

The function project_q projects the pose onto the workspace permissible by the active constraints. This metric calculates the average distance between the demonstrated poses of the given segment and those same poses constrained by the active constraints.

Similarly, we define a second metric L_{IM+AC} which considers the demonstrated data as a sequence of displacements between consecutive poses $\boldsymbol{H}_{\mathcal{E}}(n)$ and $\boldsymbol{H}_{\mathcal{E}}(n+1)$. This metric takes as input a known state q from an SCT and its associated input mappings and active constraints. For each demonstrated displacement, it checks whether the user could have commanded this displacement while state q was active. If not, it identifies the most similar commanded displacement. Hence, this metric evaluates how closely the demonstrated data can be reproduced under the input mapping and active constraints of state q. For example, applying this metric to the data used to learn the 'Approach' state, with the 'Approach' state itself as the reference, will result in a zero or near-zero cost. Using data from tasks requiring completely different motions, positions, or orientations will yield a high cost.

Computing this cost involves providing the optimal commands to $step_q$ (see subsection 4.3.2) to reproduce the demonstrated trajectories as closely as possible while adhering to the constraints of state q. However, these optimal commands are unknown, and the function $step_q$ is generally non-invertible, i. e. there is no direct way to determine the command responsible for a specific displacement between consecutively recorded poses. To address this, the optimal command to get the closest constrained displacement is found using an Evolution Strategy (ES) algorithm applied to the command input. The ES algorithm uses the best command from the computation of the last displacement $\boldsymbol{u}^*$ (initialized as [0,0,0] for the first displacement) to initialize its current solution, retaining only the best-found command at each step. This process is outlined in Algorithm 3.

A classic black-box optimization algorithm such as CMA-ES could have been used. The proposed simple black-box algorithm is efficient due to the low DoF of the input, all the input dimensions having the same order of magnitude, and the low complexity of $step_q$. From this we can define $L_{IM+AC}(s,q)$:

$$L_{IM+AC}(s,q) = \frac{1}{N_s}\sum_{n=1}^{N_s} Dist(\boldsymbol{H}_{\mathcal{E}}(n), \boldsymbol{H}closest\_constrained\_pose_n) \tag{6.5}$$

Those two metrics provide a similarity measure for each segment s of the demonstrated trajectories, comparing them to the states of learned SCTs. An example of this is presented in Table 6.2. Those metrics guide the skill designer in deciding whether to associate a known state q to a newly demonstrated segment. If so, components such as the input mappings and robot parameters (e. g. end-effector finger configuration)

Algorithm 3 Evolution Strategy: Computes the optimal command to find the closest displacement from $\boldsymbol{H}_{\mathcal{E}}(n)$ to $\boldsymbol{H}_{\mathcal{E}}(n+1)$, constrained by the input mappings and active constraints of state q.

Input: $\boldsymbol{H}_{\mathcal{E}}(n), \boldsymbol{H}_{\mathcal{E}}(n+1), \boldsymbol{u}^*, \text{step}_q$
Output: $\text{closest\_constrained\_pose}_n, \boldsymbol{u}^*$

$\sigma = 0.2, \boldsymbol{u} = \boldsymbol{u}^*$
while not_converged **do**
  $\text{solutions} = [\text{sol}_1, \text{sol}_2, ...],\ \text{sol}_i \sim \mathcal{N}(\boldsymbol{u}, \sigma)$
  for sol **in** solutions **do**
    $\text{cost}_{\text{sol}} = Dist(\text{step}_q(\text{sol}, \boldsymbol{H}_{\mathcal{E}}(n)), \boldsymbol{H}_{\mathcal{E}}(n+1))$
  end for
  $\boldsymbol{u} = \text{argmin}(\text{cost})$
  if successful_mutations_ratio < 0.2 **then** ▷ Ratio of commands better than the last best $\boldsymbol{u}$
    $\sigma = \sigma * (1 - 0.6)$
  else
    $\sigma = \sigma * (1 + 0.6)$
  end if
end while
$\boldsymbol{u}^* = \boldsymbol{u}$
$\boldsymbol{H}closest\_constrained\_pose_n = \text{step}_q(\boldsymbol{u}^*, \boldsymbol{H}_{\mathcal{E}}(n))$

from q are transferred to a newly created state q' based on the segment s. The active constraints can also be transferred or retrained on the new data using the same model.

In summary, these two metrics are defined independently of the models used for skill representation, promoting the reuse of states from relevant known SCTs.

6.4 Evaluation

This evaluation aimed to demonstrate that the extracted SCTs can be used for successful task completion. We constructed SCTs for two tasks: the first one, *open drawer*, was learned uniquely from the demonstrations and used to initialize an empty SCT library. The second, *open cabinet*, showcased an example of SCT transfer.

6.4.1 Learning to open a drawer from demonstrations

Starting with an empty SCT library, we initially learned the SCT *open drawer* from three kinesthetic demonstrations. The trajectories were automatically segmented on high curvature points, producing four segments per trajectory, as shown in Fig. 6.3.B. Various models were then fitted on those segments with $M = 20$, $\alpha = 0.2$, and $\beta = 0.4$, yielding the results presented in Table 6.1.

The skill designer then selected different models: a cone for the first state to guide the user to a pre-grasp pose while allowing significant freedom of motion and prismatic constraints for making contact with the drawer and pulling it. Prismatic constraints were selected instead of a line despite its higher cost due to the designer knownledge of the task that the orientation should also be constrained. Finally, the transitions were defined, and scaled translational control was selected as input mapping, with the scaling depending on the state. The final skill representation is shown in Fig. 6.4 and Fig. 6.5, *top*.

6.4.2 Learning to open a cabinet door from demonstrations and a known SCT *open drawer*

The second task of interest was opening a cabinet door. One demonstrated trajectory was recorded with kinesthetic teaching and then automatically segmented. Due to the similarity between the two tasks, the resulting segments matched the ones used to learn the skill *open drawer*. Subsequently, for each segment

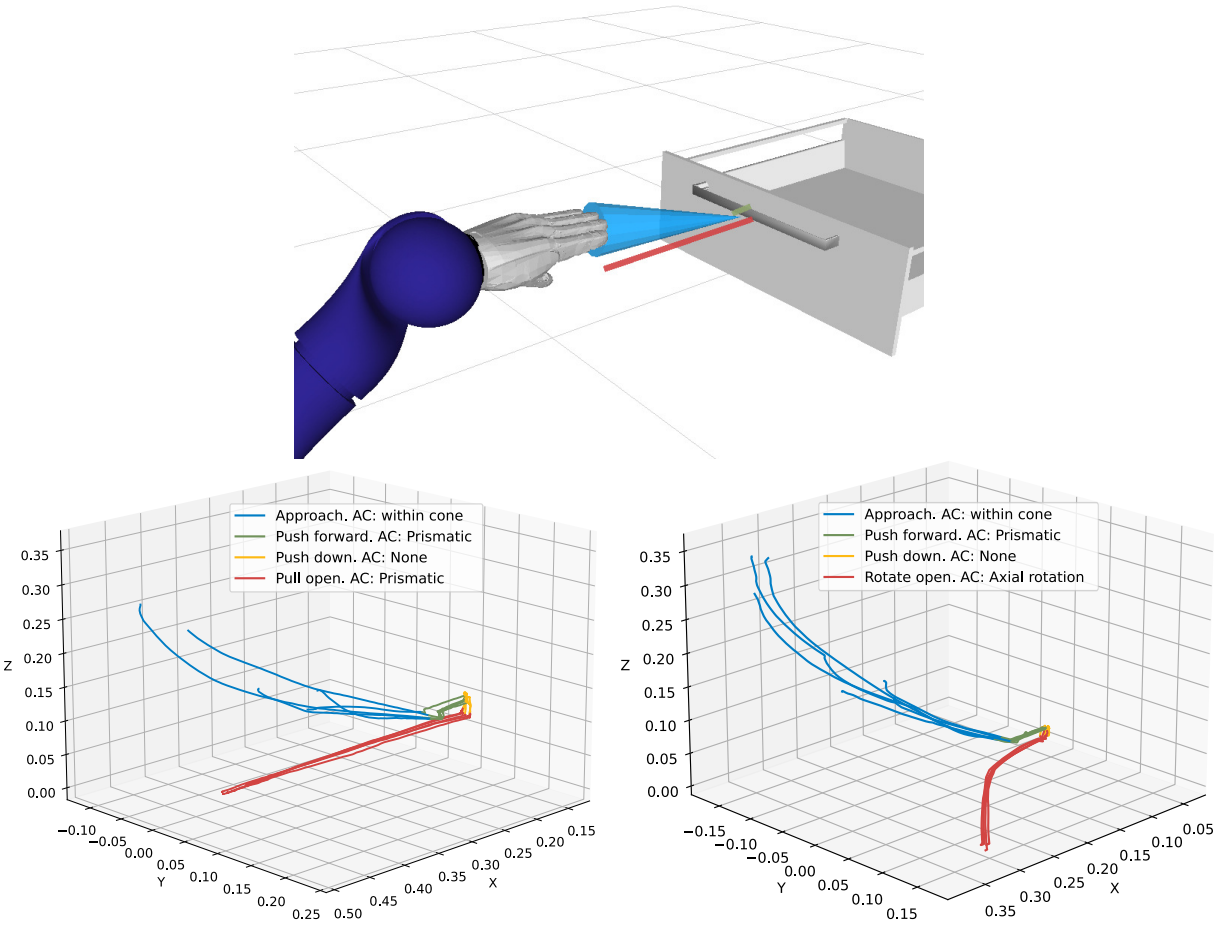


Figure 6.5: **Learned SCTs.** We show a visualization of the learned *open drawer* SCT (top), and the measured trajectories of five executions of opening a drawer (bottom left) and a cabinet door (bottom right) with a 3 DoFs joystick with EDAN. The colors illustrate the four states of each SCT. The two SCTs differ only in the fourth state, where an axial rotation constraint is used for *open cabinet door* instead of the prismatic motion of *open drawer*.

of the demonstrated trajectory and each state of *open drawer*, L_{AC} and L_{IM+AC} were computed, yielding the results shown in Table 6.2.

Guided by these results and their prior knowledge, the skill designer built the first three states of *open cabinet* as copies of the SCT *open drawer*, as they corresponded to grasping an identical handle (see Fig. 6.2.B), with their associated transitions. Notably, as highlighted in bold in Table 6.2, there were clear similarities between the first segment and the 'Approach' state and between the second segment and the 'Push forward' state. However, the fourth segment did not correspond to any existing state. It was assigned a new constraint, modeled as an *axial rotation* to restrict movement in the horizontal plane and constrain the end-effector orientation.

6.4.3 Successful task completion with learned SCTs

To demonstrate that the SCTs extracted from the demonstrations enabled users to execute tasks even with low-dimensional command signals, an able-bodied expert successfully performed five executions of opening both the drawer and the cabinet door while operating EDAN. Command inputs were generated with a 3 DoFs joystick. As in previous chapters, the user controlled the end-effector while the wheelchair

Constraint class	Constraint	Cost Approach	Cost Pull
Parameterized Surfaces	plane	0.022	0.001
	line	0.051	0.004
	planar	0.035	0.004
	arc circle	0.051	0.008
	prismatic	0.060	**0.013**
	axial rotation	0.063	0.031
Parameterized Volumes	cone	**0.002**	0.0006
	curved funnel	0.020	0.009
	cylinder	0.035	0.012

Table 6.1: SCT *open drawer* models fitting results for the first segment. The selected models are highlighted in bold.

Segment \ State	Approach	Push forward	Push down	Pull
1	L_{AC}: **0.022** L_{IM+AC}: **0.013**	L_{AC}: 0.061 L_{IM+AC}: 0.057	L_{AC}: 0 L_{IM+AC}: 0.001	L_{AC}: 0.08 L_{IM+AC}: 0.078
2	L_{AC}: 0.039 L_{IM+AC}: 0.030	L_{AC}: **0.005** L_{IM+AC}: **0.005**	L_{AC}: 0 L_{IM+AC}: 0.002	L_{AC}: 0.030 L_{IM+AC}: 0.031
3	L_{AC}: 0.054 L_{IM+AC}: 0.044	L_{AC}: 0.013 L_{IM+AC}: 0.017	L_{AC}: 0 L_{IM+AC}: 0.001	L_{AC}: 0.016 L_{IM+AC}: 0.013
4	L_{AC}: 0.153 L_{IM+AC}: 0.1	L_{AC}: 0.082 L_{IM+AC}: 0.084	L_{AC}: 0 L_{IM+AC}: 0.003	L_{AC}: 0.062 L_{IM+AC}: 0.064

Table 6.2: **Comparison of a newly recorded *open cabinet* trajectory to the SCT skill *open drawer*.** In bold, the relevant results for components transfer, as states 'Approach' and 'Push forward' have low costs. Costs from state 'Push down' are negligible as there are no active constraints. L_{AC} and L_{IM+AC} are similar because the default input mapping is used for each state in this example, but this could change significantly depending on the chosen input mapping. Costs are high in 'Pull' as the trajectories are different, requiring to fit new constraints.

automatically adjusted its position through whole-body impedance control, see subsection 3.4.1. The execution of those tasks is shown in Fig. 6.6.

The constraints of the skill *open drawer* and the trajectories generated by executing the learned SCT are displayed in Fig. 6.5, *bottom* as well as in the supplementary video linked to [Que+21]. The trajectories illustrate that some constraints, such as the approach cone, provide more freedom of movement (e.g. when approaching the drawer handle), while others impose stricter limitations (e.g. the prismatic motion required to open the drawer).

6.5 Discussion

The method presented in this work accelerates the design of SCTs compared to the purely manual design presented in previous chapters, particularly by aiding the skill designer in one of the most time-consuming aspects of an SCT creation: constraint design. This is achieved by fitting and evaluating multiple models, as well as offering metrics that leverage a library of existing skills to speed up the design of new SCTs.

The calculated costs for the different models may raise the question of why the best model cannot be automatically chosen based on these costs. This can be explained using, as an example, the trajectory segments for pulling open a drawer. Although a plane has the lowest cost in this scenario, a prismatic

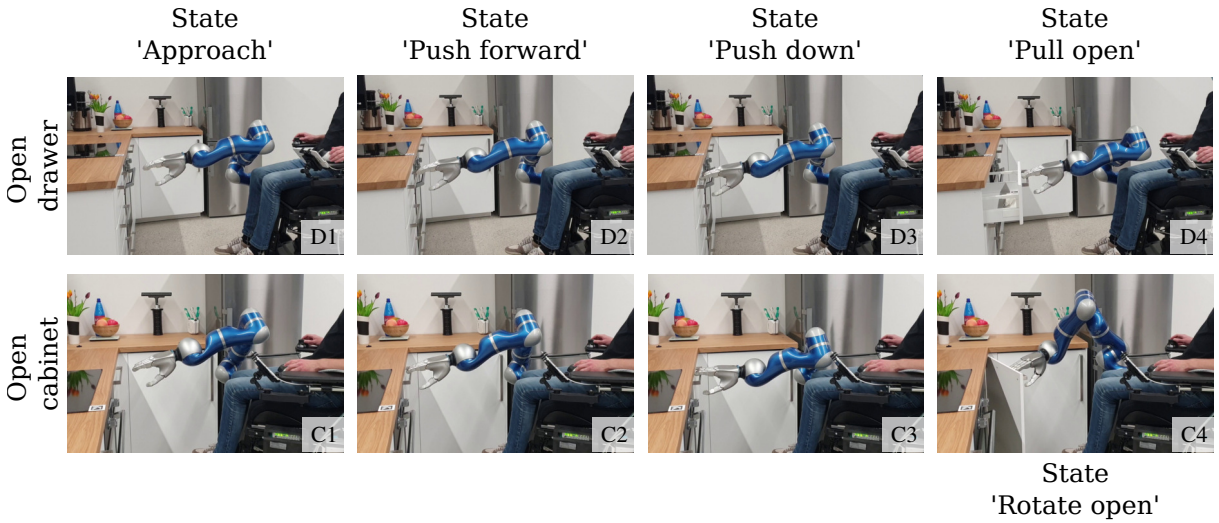


Figure 6.6: **Photo series of the different states of *open drawer* (D1-D4) and *open cabinet door* (C1-C4).** The user sitting in EDAN controls the robot with a 3D joystick. **D1 & C1** 'Approach' The robotic manipulator is restricted within a cone to guide the user towards the target handle. **D2 & C2** 'Push forward' After reaching the pre-grasp pose, the user gets to the drawer handle through a prismatic motion. **D3 & C3** 'Push down' Transition to the next state is upon contact with the handle. **D4** 'Pull open' A prismatic constraint leads the robot to pull open the drawer robustly. **C4** 'Rotate open' An axial rotation constraint is in effect to open the cabinet door.

motion arguably constrains the phase more effectively. Here, the expertise of the skill designer comes into play. Further refinement of the cost metric, such as incorporating constraints DoFs or negative examples, could help enhance automation.

While the current number of model classes is limited, the underlying methods can be applied to a broader range of models. Any methods fitting the definition of input mappings or active constraints can be used. Notable models include those that can act as active constraints, such as Task Space Regions [BSK11], Gaussian Mixture Models [ZHC18], Generalized cylinders [AC18]. Others are those that can act as input mappings, such as state-conditioned latent actions [Los+20] or a linear mapping [Prz+23].

6.6 Conclusion

This chapter presented a method to semi-automatically learn SCTs with a multi-model representation from demonstrations. An approach that allows transferring knowledge from existing SCTs to new ones was also introduced. The evaluation of a real-world scenario proved the effectiveness of our method.

The next chapter will cover another method to learn input mapping and active constraints from demonstrations, Kernelized Movement Primitives. Its formulation allows model adaptation via user commands to deal with new situations.

Chapter 7

Probabilistic learning and adaptation of active constraints

The results in this chapter have been partially presented at ICRA 2024, see [Que+24].

7.1 Introduction

As discussed in previous chapters, shared control – combining human input commands with an autonomous control system to achieve a common goal – empowers users of assistive devices with the ability to interact with their environment conveniently. Chapter 4 introduced a new framework, Shared Control Templates, which provides users with robust and transparent assistance. Chapter 5 demonstrated how this framework could be used by able-bodied and motor-impaired users for activities of daily living in different contexts, such as our laboratory, participants' homes, and events like the CYBATHLON. Currently, designing SCTs requires robotics expertise, but to fully exploit their potential, SCTs should be easier to design and modify. Chapter 6 took a first step in improving SCTs design by introducing a learning from demonstration method to learn active constraints and leverage a skill library, accelerating the design of new skills and reducing redundancy in skills design.

In this chapter, we explore probabilistic learning from demonstration to facilitate the design and adaptation of SCTs. Namely, we propose that the required constraints to manipulate objects are identified from demonstrated robot trajectories and modified via user commands when new task conditions arise, such as an environment change. Leveraging user commands from external devices to correct assistive models locally is an interesting alternative to providing new kinesthetic demonstrations or haptic feedback, particularly in settings where motor-impaired users are involved. Given their ability to approximate continuous functions, measures of uncertainty and strong adaptation properties, probabilistic learning from demonstration approaches are promising candidates to fulfill such requirements. Thus, we follow a non-parametric approach using Kernelized Movement Primitives (KMP) [Hua+19], see section 7.2, to realize the learning and adaptation of SCT constraints with the human in the loop, aiming to make the skill design intuitive and easy to use. We leverage the fact that KMP encodes the variance and correlations in the data, which we exploit to define active constraints, providing a generalized cylinder behavior [AC18] and input mappings. In addition, we build on recent results in uncertainty-aware, computationally-efficient motion primitive modulation [SH23] to adapt the learned constraints smoothly and locally, directly from user commands. The result is an interactive imitation learning approach [Cel+22] that does not require physical interactions with the robot for adaptation. Our contribution is summarized in the following:

- We introduce an approach using Kernelized Movement Primitives (KMP) [Hua+19] to derive active constraints from the variance of end-effector trajectories, for example demonstrated by kinesthetic

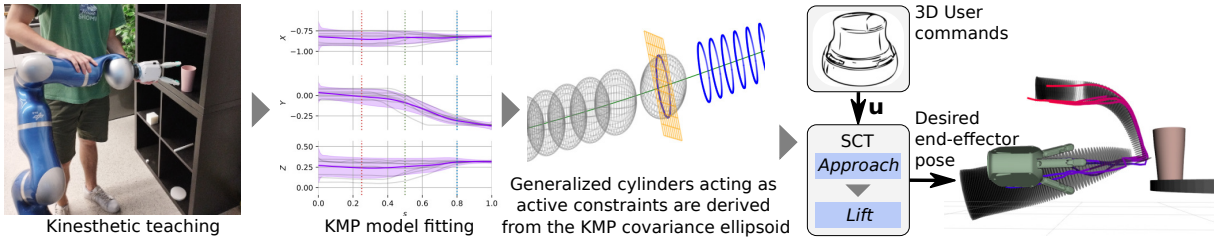


Figure 7.1: **Schema of the proposed approach to design SCTs with a probabilistic model: Kernelized Motion Primitives (KMP).** A KMP model is fitted to a set of demonstrated trajectories for each phase of the desired task. Active constraints and/or input mappings are then derived from the model and used by the skill designer to build an SCT. Finally, the robot is constrained to successfully complete the task from user commands.

teaching, see subsection 7.3.1.

- Making use of KMP adaptation capabilities [SH23], we present an *adaptation mode* allowing users to modify SCTs while using EDAN, adapting the assistance provided by the framework to new environmental constraints and requirements, such as following a different path to complete a task, see subsection 7.3.2. We show in subsection 7.3.5 how this formulation allows adjustment of the locality of the adaptation.
- Finally, we derive a scaled translational input mapping from a KMP, see subsection 7.3.6.

Our approach is experimentally validated on EDAN by performing a picking task, see section 7.4 and Fig. 7.1. We show that, using our approach, multiple able-bodied users are able to not only complete the task but also adapt the robot's assistance to new situations.

7.2 Background

7.2.1 Kernelized Movement Primitives (KMP)

KMP [Hua+19] are function approximators used in imitation learning to predict the value of an output variable $\boldsymbol{\xi} \in \mathbb{R}^D$ given observations of an input $\boldsymbol{s} \in \mathbb{R}^I$ from a set of end-effector trajectories. A KMP assumes that a *reference trajectory distribution* $\{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^N$, encoding the means, variations and correlations of $\boldsymbol{\xi}$, is available to model $\mathcal{P}(\boldsymbol{\xi}|\boldsymbol{s}_n)$, where $\boldsymbol{s}_{n=1,\dots,N}$ are N given inputs. In [Hua+19] as well as in this work, $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$ are computed from a GMM. The expectation of the output variable is computed for a test input $\boldsymbol{s}^*$ using:

$$\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{k}^* \left(\boldsymbol{K} + \lambda_1 \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\mu}, \tag{7.1}$$

where $\boldsymbol{k}^* = [\boldsymbol{k}(\boldsymbol{s}^*, \boldsymbol{s}_1), \dots, \boldsymbol{k}(\boldsymbol{s}^*, \boldsymbol{s}_N)]$, $\boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{s}_1,\boldsymbol{s}_1) & \dots & \boldsymbol{k}(\boldsymbol{s}_1,\boldsymbol{s}_N) \\ \vdots & \ddots & \vdots \\ \boldsymbol{k}(\boldsymbol{s}_N,\boldsymbol{s}_1) & \dots & \boldsymbol{k}(\boldsymbol{s}_N,\boldsymbol{s}_N) \end{bmatrix}$, $\boldsymbol{k}(\boldsymbol{s}_i, \boldsymbol{s}_j) = k(\boldsymbol{s}_i, \boldsymbol{s}_j)\boldsymbol{I}$ and $k(\boldsymbol{s}_i, \boldsymbol{s}_j)$ is a kernel function, such as the radial basis function kernel. $\lambda_1 > 0$ is a regularization factor and $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_1^\top, \dots, \boldsymbol{\mu}_N^\top\right]^\top$, $\boldsymbol{\Sigma} = \text{blockdiag}(\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_N)$, a block-diagonal matrix. Moreover, the covariance of the output is given by:

$$\mathbb{C}ov[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \alpha \left(\boldsymbol{k}^{**} - \boldsymbol{k}^* \left(\boldsymbol{K} + \lambda_2 \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{k}^{*\top}\right), \tag{7.2}$$

where α is a scaling factor, $\lambda_2 > 0$ is a regularization factor and $\boldsymbol{k}^{**}$ denotes the evaluation of the kernel function at $\boldsymbol{s}^*$, see [Hua+19] for details.

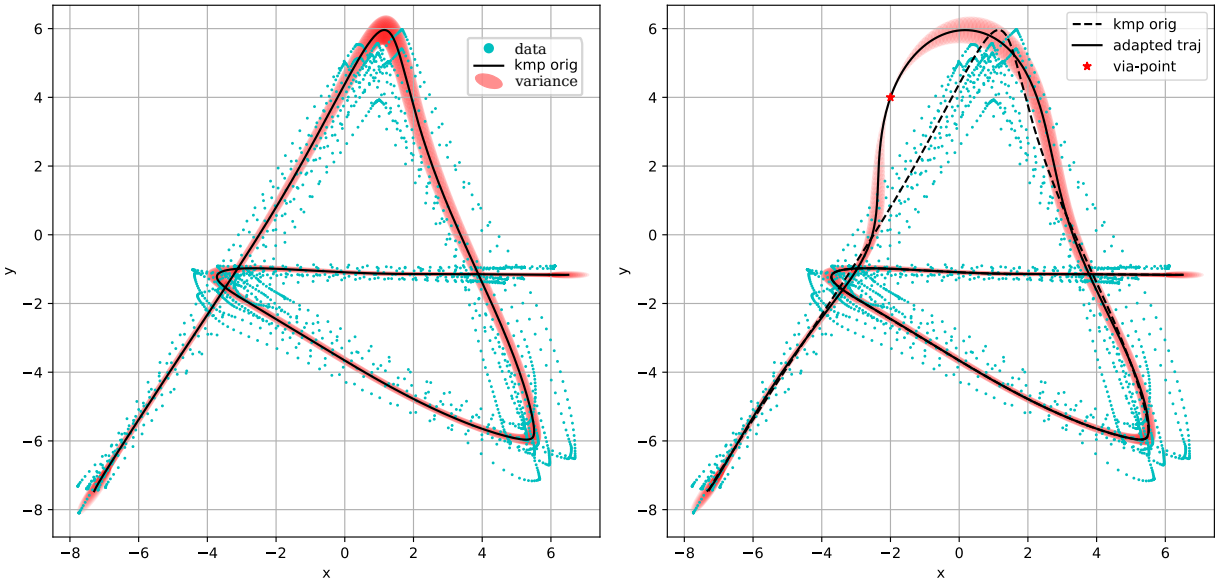


Figure 7.2: **Left:** KMP fitted on example data with mean and covariance. **Right:** Impact of adding a via-point on mean and covariance.

This model can, for example, learn position based on time, orientation based on position, or velocity based on pose. An illustration of a KMP fitted on a 2D data can be seen in Fig. 7.2, *left*.

Two methods for adapting KMP have been proposed and presented in the following sections.

7.2.2 KMP adaptation using via-points

As shown in [Hua+19], KMP provides a principled way for trajectory modulation when new task conditions arise. Particularly, for a new input $\bar{\boldsymbol{s}}$, adding the pair $\{\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}\}$ to the reference trajectory distribution will ensure that the expected trajectory passes through a desired via-point $\bar{\boldsymbol{\mu}}$, provided that $\bar{\boldsymbol{\Sigma}}$ is small enough. While this ensures adaptation, it modifies the covariance profile through $\bar{\boldsymbol{\Sigma}}$ which can have undesirable effects if the variance (Equation 7.2) is used to represent task space constraints, leading to excessively constrained motions. An illustration of adding a via-point to a KMP can be seen in Fig. 7.2, *right*. Additionally, it requires inverting the term $\boldsymbol{K} + \lambda\boldsymbol{\Sigma}$ every time a via-point is added, with $O(n^3)$ complexity.

7.2.3 KMP adaptation using null space actions

The original KMP formulation was extended by Silverio *et al.* [SH23] with a term that locally modulates the trajectory distribution, enabling a more efficient adaptation with $O(n^2)$ complexity, with no impact on the covariance profile. By defining a desired modulation $\hat{\boldsymbol{\xi}}$ at an input $\hat{\boldsymbol{s}}$ the resulting expectation is computed as:

$$\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{k}^*\boldsymbol{\Psi}\boldsymbol{\mu} + \left[\hat{\boldsymbol{k}}^* - \boldsymbol{k}^*\boldsymbol{\Psi}\hat{\boldsymbol{K}}\right]\hat{\boldsymbol{\xi}}, \tag{7.3}$$

where $\boldsymbol{\Psi} = (\boldsymbol{K} + \lambda_1\boldsymbol{\Sigma})^{-1}$ and $\hat{\boldsymbol{k}}^*$, $\hat{\boldsymbol{K}}$ are evaluations of the kernel function at $\hat{\boldsymbol{s}}$, the location where the null space action is applied. Note that the first term in Equation 7.3 corresponds to Equation 7.1, and $\boldsymbol{\Psi}$ only needs to be computed once.

The second term includes a projector $\left[\hat{\boldsymbol{k}}^* - \boldsymbol{k}^*\boldsymbol{\Psi}\hat{\boldsymbol{K}}\right]$ that modulates the original expectation only locally. This projector is referred to in [SH23] as a *soft null space projector* since, on the one hand, it

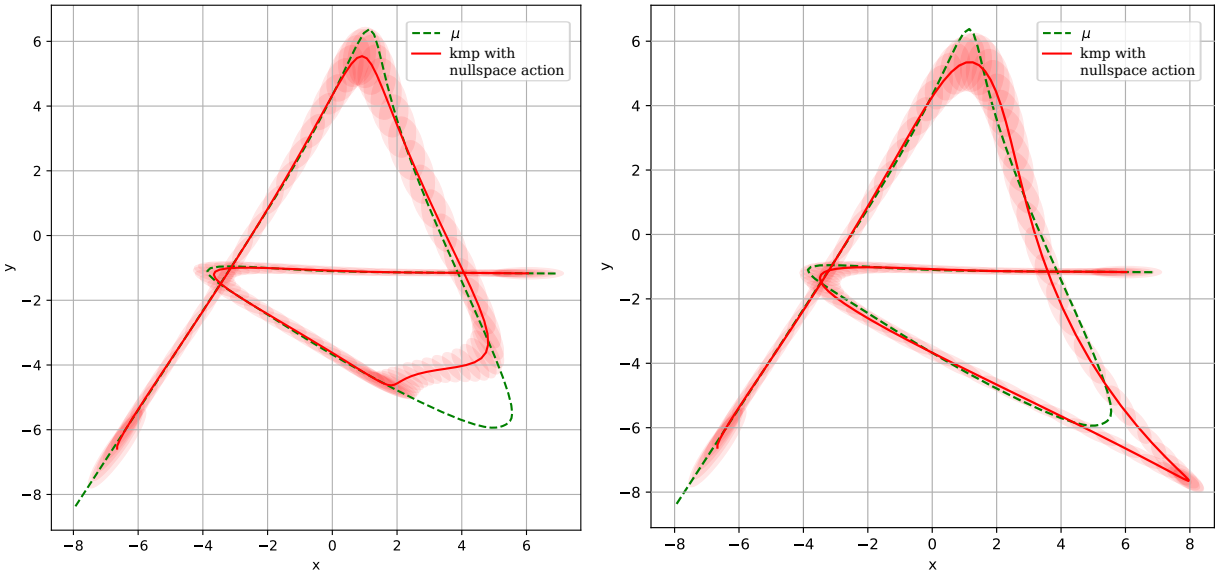


Figure 7.3: **Effect of null space actions on learned KMPs. Left:** Null space action: [-200, 100]. **Right:** Null space action: [200, -100].

is the solution to a least squares problem with null space and, on the other hand, it is a 'soft' projector, allowing $\hat{\boldsymbol{\xi}}$ to modify the original expectation in proportion to the data variance in the neighborhood of $\hat{\boldsymbol{s}}$.

7.2.4 Generalized cylinder

Another model relevant to this work is the generalized cylinder, whose application in robotics is discussed in Ahmadzadeh *et al.* [AC18]. Consider an ellipse, denoted as ρ, lying on a 2D plane perpendicular to an arbitrary smooth curve Γ in Cartesian space $\mathbb{R}^3$. The 3D surface produced by moving the plane containing ρ along the curve Γ (referred to as the directrix) while maintaining the perpendicularity of the plane to Γ forms a generalized cylinder, see [AC18] for details. Although any smooth, simple closed curve can serve as ρ, this chapter focuses exclusively on ellipses (of which the size varies along Γ).

7.3 Proposed approach

This section introduces an approach to derive active constraints and input mappings from a KMP and adapt them with external user commands when new task conditions arise. The KMP expectation (Equation 7.1) is used to define a reference path, which, in combination with the covariance (Equation 7.2) at each point, allows the derivation of a generalized cylinder behavior [AC18] that constrains the end-effector only along the directions orthogonal to the reference path (subsection 7.3.1), see Fig. 7.1 for an overview. Subsequently, we use the approach from [SH23] to modulate the learned active constraints from user commands $\boldsymbol{u} \in \mathbb{R}^D$ by computing the term $\hat{\boldsymbol{\xi}}$ directly from $\boldsymbol{u}$. Thus, we act on the KMP expectation through the projector (Equation 7.3) and consequently on the reference path (subsection 7.3.2). Fig. 7.4 illustrates this concept. To mitigate undesired correlations between actions and adjust the adaptation locality, we introduce extensions of [SH23] (subsection 7.3.4 and subsection 7.3.5). Finally, the derivation of an input mapping is shown in subsection 7.3.6.

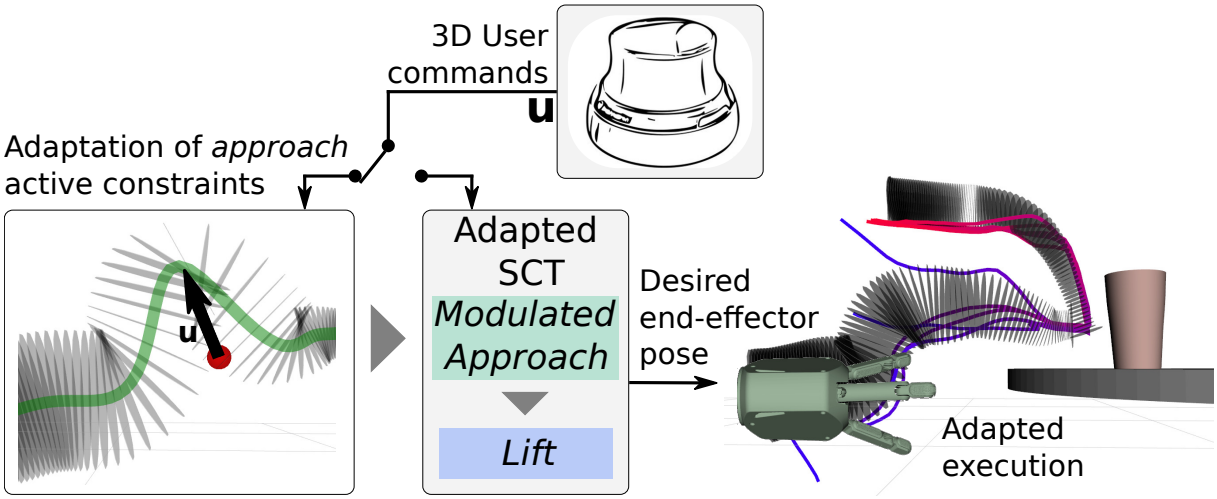


Figure 7.4: **Model adaptation.** If an SCT requires adaptation due to environmental changes or user preferences, an *adaptation mode* is entered, where the user can directly adapt the SCT with their commands. The adapted SCT is then executed.

7.3.1 Deriving active constraints from a KMP

Let us assume a set of H demonstrations with M datapoints each $\{\{\boldsymbol{s}_{m,h}, \boldsymbol{\xi}_{m,h}\}_{m=1}^{M}\}_{h=1}^{H}$ where $\boldsymbol{s} \in \mathbb{R}$ is a normalized time variable aligned across all demonstrations using dynamical time warping [SC07] and $\boldsymbol{\xi} \in \mathbb{R}^3$ is the end-effector position.

Querying a KMP model fit on such a dataset with equally spaced inputs $\boldsymbol{s}^*_{i=1,\dots,N} \in [0, 1]$ provides a trajectory distribution over end-effector positions with associated means and covariance matrices $\{\boldsymbol{\mu}^*_i, \boldsymbol{\Sigma}^*_i\}_{i=1}^{N}$. We propose to use the KMP expectation (Equation 7.1), given by the means $\{\boldsymbol{\mu}^*_i\}_{i=1}^{N}$, as the directrix Γ of a generalized cylinder. Additionally, for every point i along Γ, the covariance matrix $\boldsymbol{\Sigma}^*_i$ at an arbitrary variance threshold is treated as an ellipsoid. Its intersection with a plane passing through its center and perpendicular to the directrix – following the approach described in [Kle12] – creates an ellipse ρ, see also Fig. 7.1, *center*. The set of ellipses computed at every point of Γ, $\{\rho_i\}_{i=1}^{N}$, forms a surface with smoothly-varying cross-sections – a generalized cylinder [AC18].

A generalized cylinder can be computed offline from the KMP and can be used as an active constraint in Cartesian space, constraining the desired end-effector position within its volume. The $\boldsymbol{\mu}^*_i$ closest to the end-effector is identified, after which the end-effector is projected to the point of the cylinder volume spanned by ρ_i with the shortest distance. In practice, for better adaptation performance (see next section), we sample online the KMP at each SCT $project$ call to get ρ_i and compute the projection. The generalized cylinder is then only used for visualization.

With this constraint, a high variance (i. e. spanning a wide ellipse) at a specific section of the demonstrated trajectories provides motion freedom to the user. In contrast, a low variance constrains the user, e. g. at a specific grasp pose. Example trajectories resulting from using two KMP models are shown in Fig. 7.1, *right*. Note that using a generalized cylinder behavior guarantees unconstrained motion along the direction of Γ, unlike projecting on a full covariance ellipsoid, which may have low variance along the direction of motion. The process to build an active constraint with demonstrations and KMP is illustrated with Algorithm 4.

If the end-effector is at the beginning or the end of the directrix, an additional projection within a spherical volume – its radius the maximum radius of the associated ellipse – is enforced by default. Otherwise, the end-effector would not be constrained in the directrix direction.

Now that we have active constraints extracted from demonstrations represented as a KMP, we propose

Algorithm 4 Constraint extraction from demonstration with a KMP model

1: **Initialization**
- Set KMP hyperparameters l, λ, α

2: **Learning from demonstrations**
- Collect demonstrations $\{\{\boldsymbol{s}_{n,h}, \boldsymbol{\xi}_{n,h}\}_{n=1}^{N}\}_{h=1}^{H}$.
- Extract the reference trajectory distribution with DTW, GMM and GMR $\{\boldsymbol{s}_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^{N}$.
- Fit a KMP on the reference trajectory distribution.

3: **At run time: prediction using KMP** (see section 7.2)
- Find $\boldsymbol{\mu}_i^*$ closest to the end-effector.
- Query the KMP with $\boldsymbol{s_i}$ and $\boldsymbol{s_{i+5\%}}$ to get $\boldsymbol{\mu}_i = \mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}_i)]$, $\boldsymbol{\Sigma}_i = \mathbb{C}ov[\boldsymbol{\xi}(\boldsymbol{s}_i)]$ and the directrix direction, see Equations (7.1)–(7.2).
- Intersect $\boldsymbol{\Sigma}_i^*$ at an arbitrary variance threshold to get the ellipse ρ_i.
- Active constraint: project the end-effector within the cylinder spanned by ρ_i.

to adapt them with the user in the loop by acting directly on the KMP representation.

7.3.2 KMP adaptation with the user in the loop

Given the limitations of via-point-based KMP adaptation discussed in subsection 7.2.2, we propose to act on the null space of the KMP (Equation 7.3) and adapt the active constraint directly from user commands $\boldsymbol{u}$.

Since $\boldsymbol{u}$ usually is used to control the robot's end-effector, we propose that, when wanting to modulate the active constraints, the user can select an *adaptation mode* at any given time t_0, see Fig. 7.4. In this mode, shared control is temporarily deactivated, and user commands are interpreted as desired modifications to the underlying KMP expectation (Equation 7.1) rather than control actions for the robot. Hence, in adaptation mode, the modulation term $\hat{\boldsymbol{\xi}}$ in Equation 7.3 is calculated directly from $\boldsymbol{u}$. The impact of the user commands, i. e. the directrix deformation, is shown in RViz.

For a given location $\hat{\boldsymbol{s}}$ in the input space, where the modulation is to be applied, we define a desired correction starting at t_0 and lasting until t_1 as an accumulation of user commands:

$$\hat{\boldsymbol{\xi}}(\hat{\boldsymbol{s}}) = \int_{t_0}^{t_1} \boldsymbol{u}(t)\, dt. \tag{7.4}$$

Equation 7.4 can be used generically to define modulations to be applied on a KMP through Equation 7.3, from external user commands.

In our experiments (section 7.4), a 3D user command $\boldsymbol{u}$ is resolved into two orthogonal components: the component tangential to the directrix, $\boldsymbol{u}_{\parallel}$, and the component perpendicular to it, $\boldsymbol{u}_{\perp}$. We propose to use $\boldsymbol{u}_{||}$ to select where the modulation is taking place, $\hat{\boldsymbol{s}}$, by moving along the directrix. At the same time, $\boldsymbol{u}_{\perp}$ modulates the underlying KMP by acting on the directions orthogonal to the directrix. Note that, despite the decoupling, we have $\boldsymbol{u}_{\perp}, \boldsymbol{u}_{\parallel} \in \mathbb{R}^3$.

As the directrix continuously changes, there are two ways to compute $\boldsymbol{u}_{\perp}$ and $\boldsymbol{u}_{\parallel}$: using the original directrix or the most recent one. Using the original directrix ensures that for any $\boldsymbol{u}$, the adaptation does not affect the direction of $\boldsymbol{u}_{||}$ at a specific $\hat{\boldsymbol{s}}$. This implies that the resulting adaptation is independent of the order of the command sequence. While this option is generally intuitive for small deformations, it becomes unreliable for large deformations.

The alternative option is to use the latest directrix. However, this often leads to deformations not perpendicular to the initial directrix, which can quickly become counterintuitive.

7.3.3 End-effector displacement as null-space action

An alternative approach to using user commands as null-space actions is to adapt the KMP model based on the end-effector motion rather than directly relying on user commands. When an active skill utilizes a KMP model, the end-effector can be constrained as defined in subsection 4.3.1. However, if $\mathcal{E}_{\mathrm{IM}}$ (Equation 4.3) is displaced beyond these constraints, the distance between $\mathcal{E}_{\mathrm{IM}}$ and its projection onto the KMP constraint, $\mathcal{E}_{\mathrm{AC}}$, can be calculated:

$$\hat{\boldsymbol{\xi}}(\hat{\boldsymbol{s}}) = Dist(\mathcal{E}_{\mathrm{IM}}, \mathcal{E}_{\mathrm{AC}}) \tag{7.5}$$

The KMP model can then be updated accordingly, which will modify the constraints used in the following iteration. This allows the KMP model to act as a sort of soft constraint, which will be adjusted if the user consistently directs the end-effector against it.

7.3.4 Decorrelating adaptations

Being constructed from full covariance matrices, the projector derived in Equation 7.3 considers the correlations in the training data. In such a case, a desired modulation on a subset of the task space DoFs, e.g. $\hat{\boldsymbol{\xi}} = [\hat{\xi}_1\ 0\ 0]^\top$ in our 3D case, may affect other DoFs. While this may be a desirable feature in exploration [SH23], it may lead to unintended effects in shared control scenarios, such as deforming the directrix in a direction unwanted by the user. Therefore, we propose to decorrelate the result of applying Equation 7.4.

To this end, $\boldsymbol{u}_\perp$ is decomposed along its D components[1] $\{u_{\perp_j}\}_{j=1}^{D}$ and Equation 7.4 is computed for each component as $\hat{\boldsymbol{\xi}}_j$, with zero entries except at index j. The removal of correlations in the final modulation is then performed by computing

$$\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{k}^*\boldsymbol{\Psi}\boldsymbol{\mu} + \sum_{j=1}^{D} \bar{\boldsymbol{S}}_j \left[\hat{\boldsymbol{k}}^* - \boldsymbol{k}^*\boldsymbol{\Psi}\hat{\boldsymbol{K}}\right] \hat{\boldsymbol{\xi}}_j, \tag{7.6}$$

where $\bar{\boldsymbol{S}}_j = \mathrm{blockdiag}(\boldsymbol{S}_{j,1}, \ldots, \boldsymbol{S}_{j,N})$ is a block-diagonal selection matrix formed of D-dimensional matrices $\boldsymbol{S}_{j,i}$, with zero entries except at (j, j) which is 1.

In this manner, the uncertainty-aware modulation properties of the original projector (Equation 7.3) are preserved – regions with low or high variance require fewer or more modulation commands to be adjusted – while ensuring that actions along one DoF do not interfere with other DoFs. This effect is shown in Fig. 7.10. Alternatively, by removing off-diagonal terms, correlations could be removed directly from $\{\boldsymbol{\Sigma}_n\}_{n=1}^{N}$. However, this would also affect the ellipsoids, hence the active constraint, making the overall framework less expressive.

7.3.5 Adaptation with multiple actions

The ability to modulate the trajectory at multiple points from the same user command helps guarantee the smoothness of the resulting constraint and a lower burden for the user. Hence, we further modify Equation 7.3 to allow the application of P corrections from one single action $\boldsymbol{u}$. For example, from the original $\hat{\boldsymbol{\xi}}$ computed with Equation 7.4, a set of P desired modulations $\{\hat{\boldsymbol{\xi}}_i\}_{i=1}^{P}$ can be computed with the magnitude of each action decaying linearly from the center $\hat{\boldsymbol{\xi}}_{\frac{P-1}{2}}$ and applied on $\hat{\boldsymbol{s}}_1, \ldots, \hat{\boldsymbol{s}}_P$.

Note that applying multiple actions does not invalidate the decorrelation strategy. Their effect on the

[1] $D = 3$ in this work but the proposed formulation is general

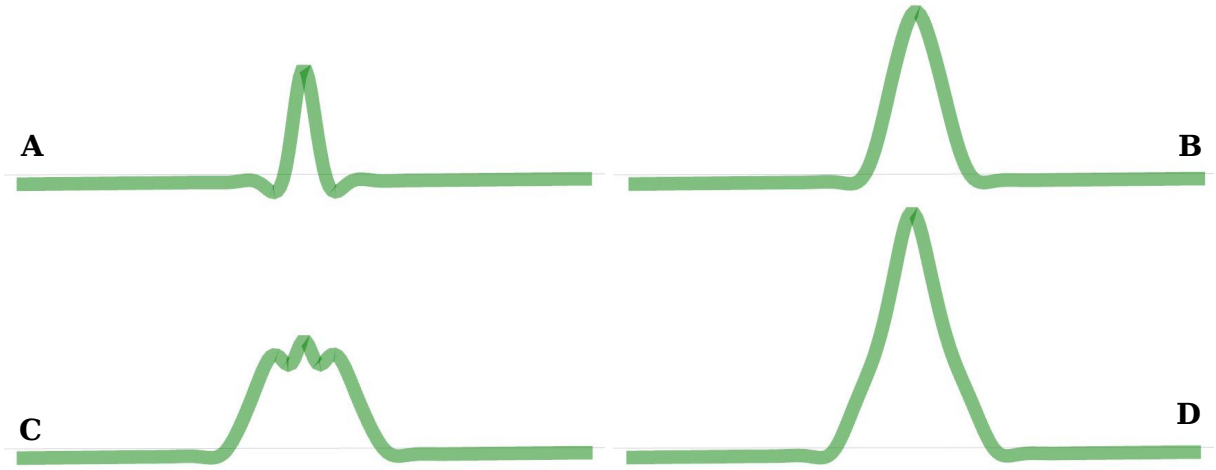


Figure 7.5: **Impact of different sets of P on a rectilinear directrix with a Matérn kernel. A**: Single action: P = 1. A slight interference at the base of the deformation is visible. **B**: P = 9. Indexes of the additional actions, in relative percentage from the original action: [0.02,0.04,0.06,0.08], with scaling: [0.7,0.6,0.3,0.1] **C**: P = 11. Indexes: [0.04,0.06,0.08, 0.10,0.12] with scaling [0.6, 0.4,0.3,0.2,0.1]. **D**: P = 7. Indexes: [0.03, 0.06, 0.09] with scaling [0.8, 0.7, 0.6].

.

original KMP can be computed cumulatively as[2]

$$\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{k}^*\boldsymbol{\Psi}\boldsymbol{\mu} + \sum_{i=1}^{P}\left[\hat{\boldsymbol{k}}_i^* - \boldsymbol{k}_i^*\boldsymbol{\Psi}\hat{\boldsymbol{K}}_i\right]\hat{\boldsymbol{\xi}}_i. \tag{7.7}$$

Removing the correlations between different DoFs (subsection 7.3.4) is achieved by combining Equation 7.6 and Equation 7.7, yielding

$$\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{k}^*\boldsymbol{\Psi}\boldsymbol{\mu} + \sum_{i=1}^{P}\sum_{j=1}^{D}\bar{\boldsymbol{S}}_j\left[\hat{\boldsymbol{k}}_i^* - \boldsymbol{k}_i^*\boldsymbol{\Psi}\hat{\boldsymbol{K}}_i\right]\hat{\boldsymbol{\xi}}_{i,j}. \tag{7.8}$$

Equation 7.8 implements the modulations of a KMP at multiple points simultaneously while removing undesired effects between task space DoFs resulting from correlations in the training data. Note that the computational complexity[3] of Equation 7.8 is $\mathcal{O}(pn^2)$. Consequently, as $P \rightarrow N$, the complexity approaches that of the original via-point-based adaptation approach, $\mathcal{O}(n^3)$. However, this is unlikely to occur in practice, as modulations do not require to be added at every sample point around $\hat{\boldsymbol{s}}_{center}$ to have an impact. It is actually favorable to space and scale them appropriately, providing different deformation patterns. Three patterns can be seen in Fig. 7.5, next to a single action deformation. The top right pattern is a good deformation obtained empirically, while the others were selected for illustrative purposes. Any deformation can be used as a pattern: Fig. 7.6 shows examples of successive deformations, which could be used as a set of P. In practice, a skill designer would simply select the desired pattern to generate the associated P deformations for each of their actions.

Applying multiple deformations P in the proposed manner helps mitigate overfitting caused by the deformation action (which can occur for certain choices of kernel and kernel parameters), by smoothing the trajectory points away from the center.

[2]A term $\underline{K}^{-1}$ multiplies $\hat{\boldsymbol{\xi}}$ on the left in the original formulation when multiple actions are applied. However, it correlates

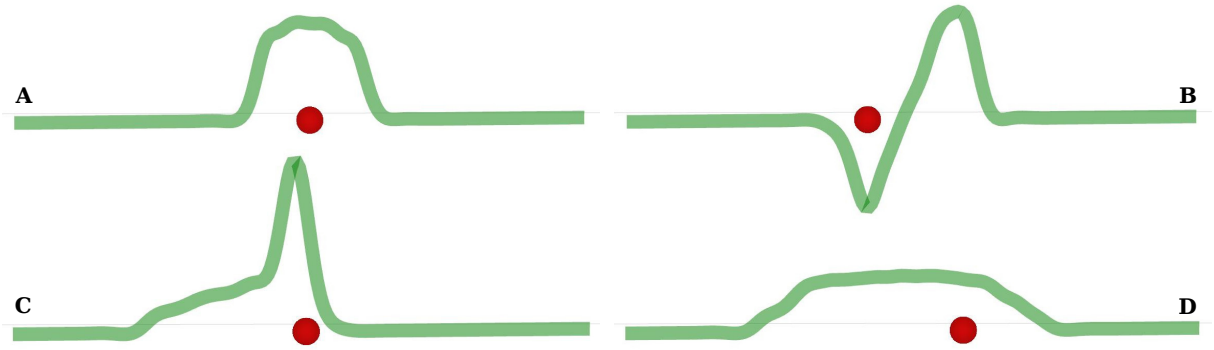


Figure 7.6: **Arbitrary deformations to illustrate the method flexibility.** The red dot indicates the point on the original directrix where the deformation is currently applied. It was moved along the trajectory to create those deformations, see Fig. 7.8. **A**: A smooth, wide deformation. **B**: Deformations in two different directions. **C**: Low amplitude deformation on one side, high amplitude on the other. **D**: A wide deformation with low amplitude.

.

7.3.6 Deriving an input mapping from a KMP

One can also derive an input mapping from a KMP model, which can be used by itself or concomitantly with an active constraint derived from the same KMP, as described above. We aim to adjust the scale of the commands based on their direction to support commands going along or towards the directrix and hinder commands going perpendicularly away from it. Intuitively, scaling down commands not going along or towards the directrix will lower the impact of undesired commands (the frequency and amplitude of which depend on the interface and user control capabilities) while not restricting the reachable space. In a similar manner as with an active constraint, this is realized by first computing an ellipse at the current $\hat{\boldsymbol{s}}$.

The scaling is based on the size of the ellipses. As before, we have $\boldsymbol{\mu}_i^*$ closest to the end-effector position, with its associated ellipse ρ_i. Suppose $u_\perp$ is going away from $\boldsymbol{\mu}_i^*$. In that case, it should be scaled down proportionally to the distance between the end-effector and the surface of the cylinder spanned by ρ_i, with an arbitrary minimum scaling λ_{min}, used when the end-effector goes beyond the cylinder:

$$\lambda_{IM} = (1 - \lambda_{min}) \frac{D(\mathcal{E}_{\mathrm{SCT}}, \mathcal{E}_{\rho_\mathrm{i}})}{D(\boldsymbol{\mu}_i^*, \mathcal{E}_{\rho_\mathrm{i}})} + \lambda_{min} \tag{7.9}$$

with $\mathcal{E}_{\rho_\mathrm{i}}$ the projection of $\mathcal{E}_{\mathrm{SCT}}$ on the cylinder surface spanned by ρ_i.

Hence, the end-effector slows down more and more when commanded towards the cylinder surface and can traverse it at minimal speed. This is illustrated in Fig. 7.7. As for active constraints, an input mapping can also be derived from an adapted KMP model.

7.4 Results

We conduct three experiments to validate our contribution: first, we learned an SCT from demonstrated data, then adapted it to new environmental conditions (new target pose, then new target object), and finally,

modulations at the action level and increases the complexity, with limited benefits for our current use case. Hence, we choose to treat individual actions independently.

[3]We neglect the effect of D as the projector does not depend on j.

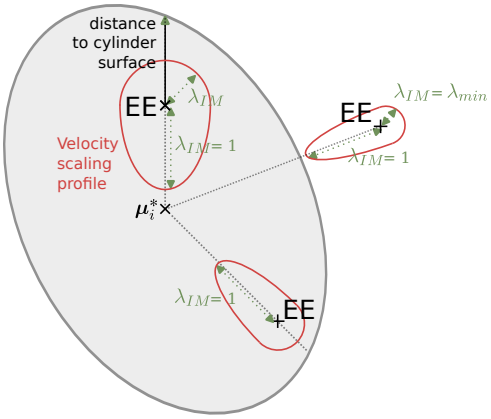


Figure 7.7: **Depiction of an input mapping scaling λ_{IM} from a cylinder**. Going closer to the cylinder surface slows down the end-effector while going towards the directrix is done at normal speed. This biases the end-effector to move along the directrix while leaving the whole task space accessible if the user wishes to depart from the demonstrated trajectories.

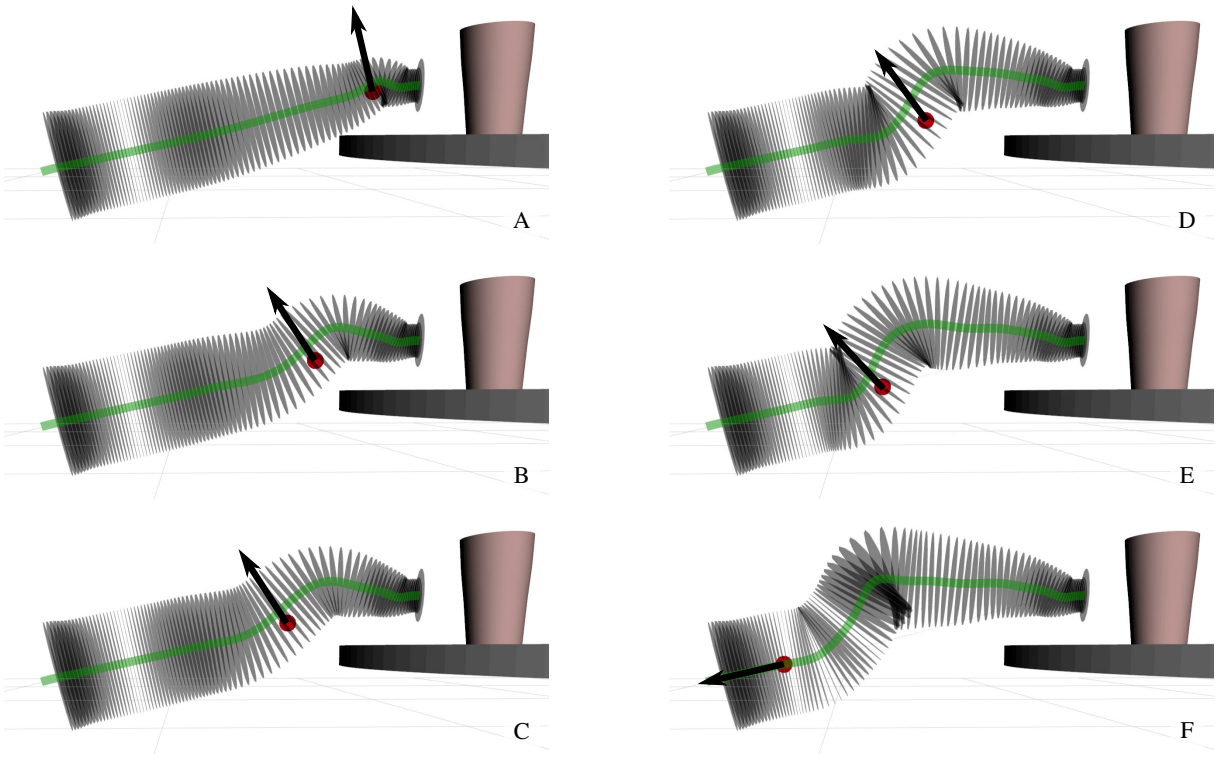


Figure 7.8: Photo series of the iterative modulation of a learned approach constraint, to ensure the fingers of the end-effector will not collide with the table with the new cup placement. User commands are depicted with a black arrow.

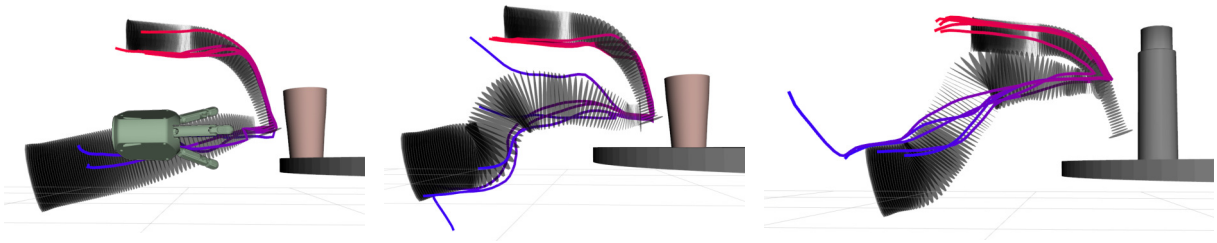


Figure 7.9: **Executions of multiple *Pick* task on EDAN with SCTs with learned constraints, with end-effector trajectories going from blue to red.** In grey, the generalized cylinders illustrate the KMP active constraints. **Left**: *Pick* SCT obtained by fitting a KMP to demonstrations. **Center**: *Pick* SCT adapted with the proposed method (in the 'Approach' state) such that the end-effector fingers do not hit the table. **Right**: A second adapted SCT, to pick up a taller object. As seen on some trajectories, the SCT execution may start with the end-effector outside of the constraint, getting smoothly pulled into the constraints due to the velocity limits, see subsection 4.3.2.

we let able-bodied expert users adapt an SCT on their own. Successful task executions on EDAN are shown for each experiment.

7.4.1 Learning active constraints for a picking skill

The first experiment aimed to pick up a cup from a table. To this end, we designed an SCT with three states: 'Approach', 'Grasp' and 'Lift'. The user triggered the transition from 'Approach' to 'Grasp' with a button click. 'Grasp' transitions to 'Lift' as soon as the grasp is done. For both the 'Approach' and 'Lift' states, five demonstrations were recorded by kinesthetic teaching. The number of demonstrations was empirically chosen: five is diverse enough in this use case. Dynamical time warping on positions was then used to temporally align the demonstrated data. A KMP was fitted to the data using a GMM with two components to compute $\{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^{N}$, with $N = 100$, $\lambda_1 = 0.25$, $\lambda_2 = 0.25$, $\alpha = 0.75$, and a Matérn kernel [RW06] with $p = 2$ and length scale $l = 0.2$, chosen empirically. The KMP was used as an active constraint, with a standard deviation of 5 to select the covariance ellipsoids. The standard deviation threshold – chosen empirically – impacts the width of the ellipses, impacting the available task space; hence, it is considered a skill parameter. A generalized cylinder was derived for visualization and can be seen in Fig. 7.9, *left*.

The orientation was set as the orientation of the closest pose from the mean trajectory obtained with dynamical time warping. A translational input mapping was added in those states. The resulting SCT is started by an expert user via the GUI, and five successful executions of the task are shown in Fig. 7.9, *left*.

7.4.2 Adapting learned active constraints to new conditions

For the second experiment, the cup was placed closer to the center of the table to introduce a change in the environment. The (object-centric) trajectory had to be adapted so that the fingers would not collide with the table when using an approach trajectory with a low height. This could be achieved by deforming the KMP, constraining the position in the 'Approach' state in adaptation mode, shown in Fig. 7.8. In this simple scenario with a precise input device, $P = 1$ was chosen. The red sphere indicates where the modulation is taking place, i. e. $\hat{\boldsymbol{s}}$, selected with $\boldsymbol{u}_{||}$. To make it easier to select $\hat{\boldsymbol{s}}$, $\boldsymbol{u}_{\perp}$ is nullified if $\|\boldsymbol{u}_{||}\| > \|\boldsymbol{u}_{\perp}\|$, so that no modulation would be applied if the user predominantly intended to move along the directrix. On exit of the adaptation mode, the model was updated, and the SCT reloaded. Then, five pick tasks were performed successfully, see Fig. 7.9, *center*.

In a third experiment, a thermos was used as the target object. It required a higher grasp, therefore

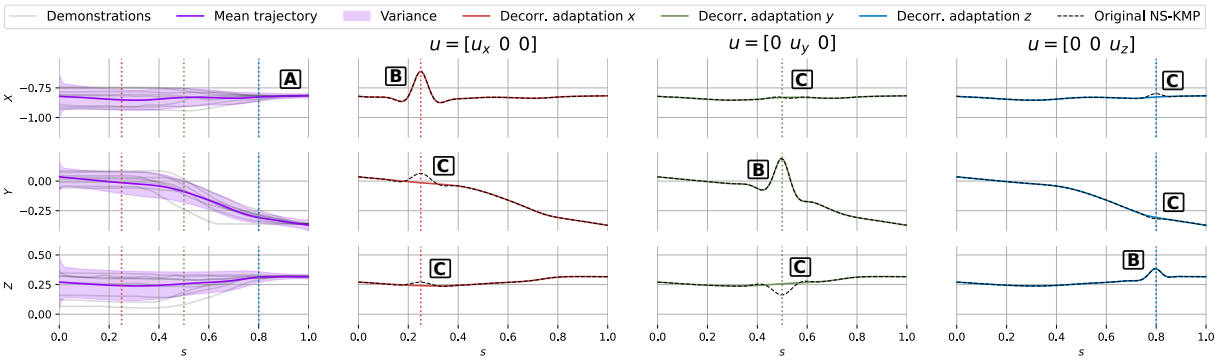


Figure 7.10: **Null space action and decorrelation adaptation.** On the left are recorded trajectories to approach a cup plotted in their x, y and z components, together with a fitted KMP model represented by its mean and variance. The variance decreases when getting closer to the target item A, reducing the task space available to the end-effector. In each subsequent column, a null space action is applied to a single dimension. The impact of modulations, i. e. its amplitude scaled by the variance, can be seen at B. The effect of the proposed decorrelation adaptation can be seen at C, where a null space action in a single dimension has no effect in other dimensions. Notice the correlation effect occurring in the original NS-KMP formulation [SH23] (dashed black line) with actions on one DoF deforming other DoFs.

requiring a second adaptation. Following the same procedure, five executions are shown in Fig. 7.9, *right*. In both experiments, the KMP active constraint of the second state could be reached and did not require adaptation.

7.4.3 Action decorrelation and computational complexity

Fig. 7.10 shows the trajectories recorded for the 'Approach' state, and the resulting mean and variance of a fitted KMP model. Additionally, the impact of a null space action is shown on the original NS-KMP formulation [SH23] and our proposed approach, illustrating the effect of decorrelating the null space actions.

Fig. 7.11 shows the computation time resulting from using null space actions or via-points to modulate the main trajectory. Two approaches to adding via-points are evaluated: either increasing the set of points of the reference trajectory distribution or replacing points (the closest ones), keeping the set size constant. As expected, null space actions are efficient as long as $P << N$.

7.4.4 Evaluating performance of new users

Finally, we evaluated how five other able-bodied expert EDAN users would modulate the original trajectory in the same setting as the first part of subsection 7.4.2: pick a cup placed closer to the table's center. They had no experience with the proposed method but are familiar with SCTs and using a Spacemouse. After explaining the SCT behavior, they tried the original SCT two to four times, as in the first experiment. Then, they practiced modulating the KMP model. At their convenience (in practice, within four minutes), they signaled they were ready to start adjusting the SCT to the new environment and entered the adaptation mode to do so. Finally, they executed the adapted SCT on the robot, which was successful for each user on the first try. Table 7.1 shows the time it took each person to modulate the trajectory and the deformation applied on each trajectory. The applied modulation is measured as the deformed trajectory length divided by the original trajectory length (which is appropriate in this specific example because the original trajectory is almost a straight line).

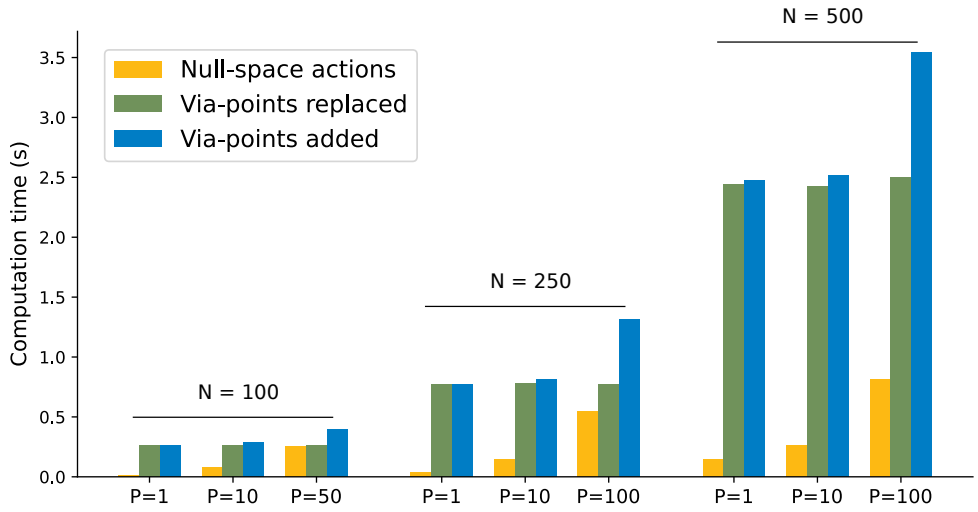


Figure 7.11: Comparison of the computation time between null space action and via-points (mean computation time over 1000 runs) for different values of P (the number of null space actions) and N (number of given inputs to a KMP). Via-points are specified to match the result of the applied null space action.

Table 7.1: Skill modulation by different users.

	Participant					
	Expert user	A	B	C	D	E
Modulation time (s)	**30**	71	87	79	40	62
Deformation %	**5**	10	34	18	10	20

7.4.5 Orientation learning

In a separate experiment using the same demonstrated data, we learned a KMP model for constraining the orientation of the end-effector based on its position. The end-effector position was used as input and the orientation represented through the exponential map as output. The KMP was then sampled every 5mm within the minimum volume 3D rectangular cuboid encompassing all demonstrated positions. This resulted in a list of pairs (position, orientation) covering the entire space where demonstrations were recorded. This data was then represented as a k-d tree [Ben75] for a fast query at run time. This provided a better representation of the orientation than using the directrix.

7.5 Discussion

7.5.1 Results analysis

The results obtained in subsection 7.4.1 and subsection 7.4.2 show that SCTs – particularly their active constraints – can be successfully learned using our proposed approach and adapted using the same input device as for task completion. These results are particularly relevant for motor-impaired users who usually cannot make physical corrections. In subsection 7.4.4, multiple able-bodied users could not only complete the task but also adapt the assistance the robot provided to a new situation by modulating the demonstrated constraints. Table 7.1 shows that, even with no prior experience with the method, this adaptation could

be done in realistic time frames (all within two minutes) and with appropriate deformations (4 out of 5 participants deformed less than 20% of the trajectory). Even if it took longer and required more deformation than the expert user, it was successful for each participant. These results suggest that the approach is intuitive and easy to use.

In addition, Fig. 7.10 validates our action decorrelation strategy, showing that individual actions on one DoF do not affect the others. This is particularly beneficial, as without it, deformations could also occur along the directrix, potentially giving it a high curvature, which is not desirable for our use case, and creating an active constraint where the user could become stuck on the 'folded' generalized cylinder surface.

We leveraged the benefits of the efficient modulation offered by Equation 7.3. A non-optimized Python implementation provided a latency low enough for online model updates, allowing intuitive modulations for a simple SCT. We did not use via-point adaptation (subsection 7.2.2), as it entails the decrease in the predicted covariance profile, which we deemed undesirable in our setting.

Using generalized cylinders directly constructed from data as active constraints is also possible. However, they tend to be less smooth than those generated using a probabilistic model. For example, sharp turns in the recorded trajectory may be clearly visible on the generalized cylinder, whereas the probabilistic approach smooths out the surface, providing better assistive behavior. Additionally, modifying a generalized cylinder directly constructed from data requires adapting individual trajectories, while a KMP provides more general adjustment capabilities with via-points and null space actions.

7.5.2 Limitations and outlook

We propose that KMP models are well-suited for learning input mappings and active constraints for SCTs, as they meet the requirements of being object-centric and time-independent, ensuring consistent shared control behaviors. Furthermore, KMP models offer flexibility for adjustments: new trajectories can always be recorded (through kinesthetic teaching or direct user control), and the model can be retrained. Additional adjustments can be made through via-points (though this reduces variance and alters the constraint) or null space actions, providing local adaptation capabilities. This provides multiple design options for the skill designer and grants users control to fine-tune assistance. SCTs' modular structure also supports hybrid approaches, where certain states can incorporate KMP-based properties while others rely on hand-coded constraints. The proposed mapping to apply adjustments is optimized for 3-DoF inputs, such as 3D joysticks or EMG-based interfaces, though other input types are also possible.

The impact of $\boldsymbol{\Sigma}$ on the adaptations, a consequence of the projector used in Equation 7.3, is particularly noteworthy as it enables more precise control in low variance sections of the demonstrations, which are likely to be the most critical for reliable task execution. However, a set of trajectories with excessive variation in $\boldsymbol{\Sigma}$ (i. e. unintended too high/low variance) may also be unintuitive for the user, as this would lead to significant variation in the speed at which the deformation happens. A formulation incorporating a hyperparameter for continuous adjustment between scaling due to $\boldsymbol{\Sigma}$ and no scaling on the null space action would be beneficial. Moreover, our approach modulates active constraints by adjusting solely the expectation of the KMP through Equation 7.3, which also affects the ellipse shape (because it affects the directrix direction). While this was sufficient for adapting a picking skill in our scenario, exploring a mechanism to modulate the covariance could be worthwhile.

There are other open avenues for research with this method. One potential direction is exploring whether contrastive learning could be integrated into the KMP formulation. For instance, negative examples like repulsive points or constraints [BDB13] could perhaps be used. This also raises the question of selecting these negative examples, such as being manually defined or derived from failed executions. Another area of interest is the representation of a skill library. Adapted skills can be stored as new skills, but skill management strategies might be necessary if frequent adaptations result in a proliferation of skills.

Further testing would be beneficial, as with the other methods discussed in previous chapters. The proposed scenario was chosen as a proof of concept to demonstrate the method's feasibility. To fully assess its effectiveness, it will be necessary to experiment with a broader variety of tasks. Additionally, input mappings generated by KMP also have to be evaluated with participants. Assessing this model for shared control faces the same challenges as those outlined in subsection 5.4.1. Evaluating constraint adaptation introduces additional difficulties:

- Users can modify the assistance itself, and effective adaptation depends on the quality of the resulting assistance, which is measured by various factors (subsection 5.4.1). As a result, no single metric objective provides a direct assessment of the quality of an adaptation.
- Furthermore, understanding users' perceptions of these adaptations could be challenging, particularly given the complexity of visualizing a 3D generalized cylinder. While RViz visualizations may be sufficient for skill designers working with a computer and Spacemouse, they are less appropriate for target users. Exploring alternative interfaces, such as AR glasses, could offer a more intuitive skill adaptation, especially when using robot motion to adjust the constraint (subsection 7.3.3).
- Finally, the current method requires a precision that not all target users posses. As with the assistance, the skill adaptation should be easy enough to use.

7.6 Conclusion

We proposed using probabilistic skill representations to learn and adapt shared control skills. We showed that using a skill representation with soft null space projectors permits a computationally efficient modulation of SCTs without decreasing motion freedom for users. Successful executions of a learned picking task, as well as the skill adaptation to new environment conditions, were shown on the assistive robot EDAN with able-bodies users. This method can help the skill designer to efficiently design new skills or adjust them to new situations. The success achieved by new users in completing the task and adapting the skill suggests that the proposed approach is intuitive and easy to use.

Chapter 8

Conclusion

This work introduced Shared Control Templates (SCT), a novel framework designed to assist users in performing activities of daily living with assistive robots. We outlined the variety of possible assistive behaviors obtained by mapping user commands and constraining the end-effector. Two approaches for learning task constraints and facilitating skill design were also introduced. The first approach involved learning a multi-model representation from demonstrations and showing a knowledge transfer from existing SCTs to a new SCT. The second approach employed a probabilistic model that can impose constraints on the end-effector and that the user can adapt to accommodate new environmental conditions. Finally, this framework was integrated into EDAN's overall control structure, allowing able-bodied and motor-impaired users to complete sequences of tasks more effectively than by using direct control.

8.1 Achievements

The goal of this work was to develop assistance that is safe, easy to use, provides agency to the user, transparent, reliable, and personalizable.

Safety was ensured through the use of a whole-body impedance controller and the implementation of virtual workspace boundaries. At the same time, the user is in control of the most meaningful motions, and the manipulator remains stationary in the absence of commands. Additionally, the assistance can be canceled at any time, allowing the user to regain complete control in direct control mode.

The system's ease of use was demonstrated through winning the 2024 CYBATHLON and the 2023 CYBATHLON Challenges, as well as through user experiments in which participants successfully performed sequences of activities of daily living.

The SCT framework was specifically designed to prioritize user agency. Input mappings can be configured to ensure that all Degrees of Freedom (DoFs) from the user commands are used intuitively and meaningfully. The robot only moves in response to user inputs, ensuring the user remains entirely in control. Additionally, SCTs can be structured so that, at any stage of a task, the user can either retrace their trajectory or, at a higher level, backtrack through the finite-state machine, providing flexibility and control throughout the task execution.

Several factors promote transparent behavior with our assistive system: separating the motion assistance from user intent estimation, using an object-centric framework, and ensuring that assistance depends solely on the current state, leading to repeatable behavior. For example, in a task such as pouring, the tilt angle of the grasped object towards a target cup remains consistent (within kinematic limits) at a fixed distance from this cup, regardless of the cup's placement, the end-effector's approach angle, or the command speed. This approach gives the user a clear understanding of the robot's state and enables intuitive control over the end-effector motion.

Reliability was achieved in three main ways. First, by limiting our scope to known objects. Second,

through the expressiveness of the methods, which enable a wide range of assistive behaviors, see Chapter 4. Any number of DoFs can be constrained using different active constraints models. In one scenario, the user might handle fine manipulation through scaled translational mapping. In another case, providing approximate forward commands is sufficient to complete the *open door* task successfully. Third, it prioritizes the user, who controls the task-relevant motions, with the ability to backtrack and try a different approach. Additionally, users can adjust the system's behavior using the "target pose correction" and "model adaptation" modes.

Personalization is achieved in several ways. First, the framework's modular design, featuring a finite-state machine structure and minimal model requirements for active constraints, provides a broad range of assistance behaviors. Multiple models can be integrated within a single skill. For example, active constraints may be manually parameterized in an initial state. In contrast, a second state could involve fitting a plane to trajectory data, while a third state could use Kernelized Movement Primitives (KMP) as an active constraint. Additionally, the hierarchical finite-state machine and object database hierarchy promote skill reuse and enhance generalization. One can also adjust the level of control provided to the user to balance agency and workload. Finally, to facilitate the design of new skills tailored to users' preferences and needs, we explored two approaches for learning from demonstrations: one using parameterized models such as surfaces and volumes, and the other a probabilistic model with a KMP. The latter also allows users to adapt a KMP constraint online.

8.2 Comparison to the state of the art

The work most closely related to ours is by Iregui *et al.* [IDA21], who propose a reconfigurable, adaptable, and modular assistance method based on Etasl (Expression-based Task Specification Language [AD14]). Their approach integrates various interfaces, estimates user intent, modulates autonomy levels, and utilizes reactive control. They also introduce "sensor interaction models", which define how the robot interacts with sensor and interface data and support a set of effects, such as applying forces, velocity, and positional adjustments. In particular, "velocity sensor interaction models" are linear input mappings. Additionally, their proposed "reactive action model" shares similarities with our active constraints model (detailed in section 4.3) and is particularly suited to dynamic constraint behaviors. In contrast, our work focuses on static constraints. Iregui's results have been demonstrated on *pick-and-place* tasks, where the skills did not require structure such as a finite-state machine. Their adaptation capabilities focus on transitioning between levels of autonomy over decoupled DoFs, whereas our approach enables constraint adaptation via direct user commands.

8.3 Limitations and future work

8.3.1 Scope of the EDAN platform and its interfaces

The SCT framework has been designed to provide assistance tailored to a user using an assistive robot such as EDAN. Input mappings and active constraints are defined in task space, which provides transparency but ignores the robot's kinematics. If the end-effector target pose $\mathcal{E}_{\mathrm{SCT}}$ brings the end-effector into a collision with the known environment, the system will disregard it, and it is up to the user to find an alternative path. However, the behavior of the manipulator's joints is not guaranteed, as its movements are reactive through the whole-body impedance controller. This issue is managed using heuristics, such as applying an elbow control action to steer away from obstacles, either predefined within a skill or calculated based on the world model. The behavior is also fine-tuned for the hardware, such as the chosen end-effector. The number of fingers and the robustness, whether direct drive or with cables, affect the type of motions and forces one can apply.

Furthermore, the SCT framework has been primarily developed with 3 DoFs interfaces in mind, such as joysticks, and particularly for noisy interfaces, such as EMG-based interfaces. Although it can also work with continuous interfaces featuring more or fewer DoFs, such as controlling the full pose with a 6D Spacemouse or a 2D joystick, this work focused on providing helpful assistance with the available EDAN interfaces. As a result, other methods may be better suited for different interfaces. Using more input DoFs should be relatively straightforward, although it increases the complexity for the user. With a single DoF, control is limited to moving along a single trajectory. By restricting the task space, most tasks we have considered can still be performed with 2 DoFs since the majority only require two relevant DoFs: one along the 'main trajectory' and another to adjust the end-effector position, for instance, with motion perpendicular to this main trajectory. For example, the main DoF might determine how much to pull open a drawer, while the secondary DoF would control the end-effector's position on the drawer handle. The challenge with using 2 DoFs is making the controls intuitive and easy to operate.

8.3.2 Evaluation

As discussed in subsection 5.4.1, evaluating shared control assistance is challenging due to the involvement of a human and differences in interfaces, robots, and environments. Even with a small sample of tasks and users, the results in Chapter 5 demonstrate that the interface plays a significant role, and mastering it is crucial for users. For this work, we focused on providing a proof of concept, showing that a system such as EDAN can enable users to perform activities of daily living. A more comprehensive evaluation of the SCT framework, exploring factors such as user enjoyment, their ability to develop an accurate mental model of the assistance and of its adaptability, and comparisons to other methods, would be highly valuable. Additionally, a long-term study involving several target group users using such a robot in their homes over an extended period would be of great interest.

As discussed in Chapter 2, numerous shared control approaches have been proposed. However, it remains challenging to discern their strengths and weaknesses from the published literature alone. Reproducing these results is difficult due to variations in hardware and software frameworks, the limited availability of open-source implementations, and the involvement of a human in the control loop. The development of simulated benchmarks is promising, see section 2.9. There, similar to reinforcement learning, multiple shared control algorithms could be applied to a range of assistive tasks. Although such simulations have limitations, they would enable straightforward comparisons across methods and facilitate testing multiple approaches on a single robotic platform. Events such as the CYBATHLON and CYBATHLON Challenges also hold great promise for advancing the field. These competitions, which require extended preparation with motor-impaired users, encourage rigorous testing, co-development, and idea exchange between teams.

8.3.3 User intent estimation and unknown objects

An SCT is designed to assist with a known task, which assumes a known target object. Furthermore, the user intent estimate is not influenced by the number, positioning, or type of objects in the environment. As a result, it is not well-suited for tasks involving many objects, such as dealing with a packed fridge. One approach to address this limitation would be considering all possible tasks when estimating user intent. Another option would be to initiate assistance when the constraints for two tasks are similar, even if the final task has not been decided upon yet, using a probability estimate for potential goals inspired by works such as [Jav+18]. This limitation has not been a significant issue so far, as the focus has been on activities like eating, drinking, moving objects, and manipulating articulated objects (e.g., doors and drawers), which typically involve only a limited number of target objects.

Currently, the system does not account for object localization precision. This factor could be incorporated into user intent inference and assistance behavior, for example, by adjusting the degree of control

based on the localization precision.

Miller *et al.* [Mil+24] have introduced an initial approach for providing assistance when grasping unknown objects. Their method combines the user's intuitive sense of an optimal approach direction with a shape completion algorithm and a grasp planner to aid in the grasping process.

8.3.4 Skill design

The current skill design process is not intended to be user-driven and relies on the skill designer, as it remains complex and not user-friendly. While constraints can be visualized and learning constraints reduce the number of states to define, there is still room for improvement. For instance, methods such as the automatic segmentation of demonstrated data and learning a finite-state machine, as done by Willibald *et al.* [WL22] (see also section 2.6), could streamline the design process. Additionally, a user-friendly graphical interface to build the event-based finite-state machine, define input mappings, active constraints, and adjust robot parameters would help. Visualization and legibility of the finite-state machine that represents the skills are important points for transparency and acceptability of the assistance by the users, and have not been investigated so far.

The decision to use either input mappings or active constraints for the same behavior is typically based on which is easier to define. However, this complicates the use of planner, which could otherwise verify in advance whether the task is achievable within the current world model. Input mappings implicitly limit the task space, while a planner requires an explicit representation of the accessible task space. Additionally, there is no current requirement for active constraints to ensure the continuity of the available task space. The task space within a skill could be structured to be connected and fully explorable to improve autonomy and the ability to perform feasibility checks. Bustamante *et al.* [Bus+24] made some initial progress in this area.

8.3.5 New skill: eating

Eating, particularly bite acquisition, is an essential daily activity that has not yet been investigated with the SCT framework. However, some research has addressed the topic with varying levels of autonomy [HHS16; Gor+23]. Methods with a higher degree of autonomy could contribute to building an SCT. For example, an SCT could provide translational control up to the point of contact with the plate, then use the type of food and the plate's state to determine how to map downwards commands: when near the plate's edge, users could control a scooping motion, while for food like spaghetti, they might control a twisting motion.

8.3.6 Manipulability

Another potential improvement is integrating manipulability metrics [Yos85] for behavior adjustment. This is particularly relevant for a system such as EDAN, which has joint limitations and a 2 DoFs null space due to its 8 joints. There are two ways this can be done: first, by optimizing the joints' configuration to increase manipulability in the main task direction, which can often be inferred by the shortest path to the next state in an SCT. Second, by assisting the user when manipulability is lacking. Let's consider the case where a user cannot make the end-effector proceed in the intended direction. For example, they pick up an apple and want to bring it to their mouth. Several options could be considered to deal with the lack of manipulability:

- If the manipulator is in a singularity (for example, when two joints are aligned), based on heuristics, a torque could be applied at one of those joints to exit the singularity.
- If it is a joint limit, reconfiguration might be necessary. Checking the self-motion manifold [Bur89] could determine whether the assistance should:

- Reconfigure the joints while maintaining the end-effector pose (i. e. remain on the same self-manifold).
- If that is not possible, reconfigure to the same end-effector pose while staying within the task space constraints, such as maintaining an upright grasped object.
- If these steps cannot provide manipulability, the system could reconfigure to the closest transition point after user confirmation.

8.3.7 Automatic SCT design

Another approach to successfully resolve tasks is to check whether they are feasible before starting them, such as investigated by Bustamante *et al.* [Bus+24]. In future work, an SCT could be derived automatically from a representation suitable for planning that incorporates constraints for articulated objects and world physics. This approach could identify where users should be given autonomy (e. g. approaching an object or choosing how much to open a drawer) and where constraints should be enforced. This has potential for tasks with 3D translational control, while it is expected to be more challenging to make it intuitive for tasks with 2D or rotational control (such as pouring).

8.4 Potential for real-life deployment

One of the key challenges in deploying assistive systems in real-world settings is ensuring they are intuitive to use. Providing reliable assistance for only a few fundamental daily activities – such as eating, drinking, or handling objects like doors, drawers, or blankets – could have a significant impact compared to current market offerings. The SCT framework addresses these needs by emphasizing user control and adapting to individual preferences while the robot's compliance ensures safety. The framework relies on a predefined object database and a competent skill designer. Overcoming these limits will be essential in bringing assistive technology to people needing it.

Bibliography

[Ben75] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[CLS79] Kevin Corker, John H Lyman, and S Sheredos. “A preliminary evaluation of remote medical manipulators”. In: *Bulletin of prosthetics research* 10.32 (1979), pp. 107–134.

[HU79] John E. Hopcroft and Jeffrey. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley, 1979. ISBN: 9780201029888.

[FB81] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[Mas81] Matthew T Mason. “Compliance and force control for computer controlled manipulators”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 11.6 (1981), pp. 418–432.

[Sho85] Ken Shoemake. “Animating rotation with quaternion curves”. In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques (SIGGRAPH)* (1985), pp. 245–254.

[Yos85] Tsuneo Yoshikawa. “Manipulability of robotic mechanisms”. In: *The international journal of Robotics Research* 4.2 (1985), pp. 3–9.

[Bur89] Joel W Burdick. “On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds”. In: *Advanced Robotics: 1989: Proceedings of the 4th International Conference on Advanced Robotics Columbus, Ohio, June 13–15, 1989*. Springer. 1989, pp. 25–34.

[Kwe+89] H. Kwee, J. J. Duimel, J. Smits, A. T. D. Moed, J. A. V. Woerden, L. W. V. D. Kolk, and J. C. Rosier. “The Manus wheelchair-borne manipulator: System review and first results”. In: *1989 IARP Proceedings, 2nd Workshop on Medical and healthcare Robotics* (1989), pp. 385–395.

[BD96] Herman Bruyninckx and Joris De Schutter. “Specification of force-controlled actions in the ’task frame formalism’-a synthesis”. In: *IEEE transactions on robotics and automation* 12.4 (1996), pp. 581–589.

[AM97] Peter Aigner and Brenan McCarragher. “Human integration into robot control utilising potential fields”. In: *1997 IEEE International Conference on Robotics and Automation (ICRA)* 1 (1997), pp. 291–296.

[Gha+98] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. “PDDL - The Planning Domain Definition Language”. In: *Technical Report, Yale Center for Computationnal Vision and Control* (1998).

[CS03] Silvia Coradeschi and Alessandro Saffiotti. “An introduction to the anchoring problem”. In: *Robotics and Autonomous Systems* 43 (2003), pp. 85–96.

[HMK03] Nikolaus Hansen, Sibylle Müller, and Petros Koumoutsakos. “Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)”. In: *Evolutionary computation* 11.1 (2003), pp. 1–18.

[AEK05] Daniel Aarno, Staffan Ekvall, and Danica Kragic. “Adaptive virtual fixtures for machine-assisted teleoperation tasks”. In: *2005 IEEE International Conference on Robotics and Automation (ICRA)* (2005), pp. 1139–1144.

[Hol+05] PJ Holliday, A Mihailidis, R Rolfson, and G Fernie. “Understanding and measuring powered wheelchair mobility and manoeuvrability. Part I. Reach in confined spaces”. In: *Disability and rehabilitation* 27.16 (2005), pp. 939–949.

[RSP05] G.R.B.E. Romer, Harry JA Stuyt, and Albér Peters. “Cost-savings and economic benefits due to the assistive robotic manipulator (ARM)”. In: *2005 9th International Conference on Rehabilitation Robotics, (ICORR)* (2005), pp. 201–204.

[TLH05] Hylke A Tijsma, Freek Liefhebber, and Just L Herder. “A framework of interface improvements for designing new user interfaces for the MANUS robot arm”. In: *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.* IEEE. 2005, pp. 235–240.

[Beh06] Sven Behnke. “Robot competitions-ideal benchmarks for robotics research”. In: *Proc. of IROS-2006 Workshop on Benchmarks in Robotics Research*. Institute of Electrical and Electronics Engineers (IEEE) New Jersey. 2006.

[RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Cambridge, MA, USA: MIT Press, 2006.

[AOH07] Alin Albu-Schäffer, Christian Ott, and Gerd Hirzinger. “A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots”. In: *International Journal of Robotics Research (IJRR)* 26.1 (Jan. 2007), pp. 23–39.

[SC07] Stan Salvador and Philip Chan. “Toward accurate dynamic time warping in linear time and space”. In: *Intelligent Data Analysis* 11 (2007).

[KSP08] Oussama Khatib, Luis Sentis, and Jae-Heung Park. “A unified framework for whole-body humanoid robot control with multiple constraints and contacts”. In: *European Robotics Symposium 2008* (2008), pp. 303–312.

[SKK08] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*. Vol. 200. Springer, 2008.

[Smi+08] Ruben Smits, Tinne De Laet, Kasper Claes, Herman Bruyninckx, and Joris De Schutter. “itasc: A tool for multi-sensor integration in robot manipulation”. In: *Int. Conf. Multisensor Fusion and Integration for Intelligent Systems*. IEEE. 2008, pp. 426–433.

[Cal09] Sylvain Calinon. *Robot programming by demonstration: A probabilistic approach*. 2009.

[Van+09] Marcel Van Gerven et al. “The brain-computer interface cycle”. In: *Journal of neural engineering (JNE)* 6.4 (2009), p. 041001.

[AK10] Panagiotis K Artemiadis and Kostas J Kyriakopoulos. “EMG-based control of a robot arm using low-dimensional embeddings”. In: *IEEE Transactions on Robotics (T-RO)* 26.2 (2010), pp. 393–398.

[DCC10] Brad E Dicianno, Rory A Cooper, and John Coltellaro. “Joystick control for powered mobility: Current state of technology and future directions”. In: *Physical medicine and rehabilitation clinics of North America* 21.1 (2010), pp. 79–86.

[Bee+11] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. "Robotic roommates making pancakes". In: *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2011, pp. 529–536.

[BSK11] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. "Task space regions: A framework for pose-constrained manipulation planning". In: *International Journal of Robotics Research (IJRR)* 30.12 (2011), pp. 1435–1460.

[Mah+11] Veronique Maheu, Julie Frappier, Philippe S Archambault, and François Routhier. "Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities". In: *2011 IEEE International Conference on Rehabilitation Robotics (ICORR)* (2011), pp. 1–5.

[Nac+11] Lennart Erik Nacke, Michael Kalyn, Calvin Lough, and Regan Lee Mandryk. "Biofeedback game design: using direct and indirect physiological control to enhance game interaction". In: *Proceedings of the SIGCHI conference on human factors in computing systems* (2011), pp. 103–112.

[VCS11] Jörn Vogel, Claudio Castellini, and Patrick van der Smagt. "EMG-based teleoperation and manipulation with the DLR LWR-III". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2011), pp. 672–678.

[Cio+12] Matei Ciocarlie, Kaijen Hsiao, Adam Leeper, and David Gossow. "Mobile manipulation through an assistive home robot". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2012), pp. 5313–5320.

[Del+12] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. "Fast approximate energy minimization with label costs". In: *International journal of computer vision* 96 (2012), pp. 1–27.

[Die+12] Alexander Dietrich, Thomas Wimbock, Alin Albu-Schäffer, and Gerd Hirzinger. "Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom". In: *IEEE Robotics & Automation Magazine (RAM)* 19.2 (2012), pp. 20–33.

[Hoc+12] Leigh R Hochberg et al. "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm". In: *Nature* 485.7398 (2012), pp. 372–375.

[Kim+12] Dae-Jin Kim et al. "How Autonomy Impacts Performance and Satisfaction: Results From a Study With Spinal Cord Injured Subjects Using an Assistive Robot". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans (T-SMCA)* 42.1 (2012), pp. 2–14.

[Kle12] Peter Paul Klein. "On the ellipsoid and plane intersection equation". In: *Applied Mathematics* 3.11 (2012), pp. 1634–1640.

[LBH12] Daniel Leidner, Christoph Borst, and Gerd Hirzinger. "Things are made for what they are: Solving manipulation tasks by using functional object classes". In: *2012 IEEE International Conference on Humanoid Robots (HUMANOIDS)* (2012), pp. 429–435.

[Phi+12] Mike Phillips, Benjamin Cohen, Sachin Chitta, and Maxim Likhachev. "E-graphs: Bootstrapping planning with experience graphs". In: *Proceedings of the International Symposium on Combinatorial Search*. Vol. 3. 1. 2012, pp. 188–189.

[BKB13] Georg Bartels, Ingo Kresse, and Michael Beetz. "Constraint-based movement representation grounded in geometric features". In: *2013 IEEE International Conference on Humanoid Robots (HUMANOIDS)* (2013), pp. 547–554.

[BDB13] Stuart A Bowyer, Brian L Davies, and Ferdinando Rodriguez y Baena. "Active constraints/virtual fixtures: A survey". In: *IEEE Transactions on Robotics (T-RO)* 30.1 (2013), pp. 138–157.

[Che+13] Tiffany L Chen et al. "Robots for humanity: using assistive robotics to empower people with disabilities". In: *IEEE Robotics & Automation Magazine* 20.1 (2013), pp. 30–39.

[DLS13] Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. "Legibility and predictability of robot motion". In: *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE. 2013, pp. 301–308.

[DS13] Anca D Dragan and Siddhartha Srinivasa. "A policy-blending formalism for shared control". In: *International Journal of Robotics Research (IJRR)* 32.7 (2013), pp. 790–805.

[Hau13] Kris Hauser. "Recognition, prediction, and planning for assisted teleoperation of freeform tasks". In: *Autonomous Robots* 35.4 (2013), pp. 241–254.

[Jai+13] Ashesh Jain, Brian Wojcik, Thorsten Joachims, and Ashutosh Saxena. "Learning trajectory preferences for manipulators via iterative improvement". In: *Advances in neural information processing systems* 26 (2013).

[Kim+13] Jeonghee Kim et al. "The tongue enables computer and wheelchair control for people with spinal cord injury". In: *Science translational medicine* 5.213 (2013).

[Nam+13] Yunjun Nam, Bonkon Koo, Andrzej Cichocki, and Seungjin Choi. "GOM-Face: GKP, EOG, and EMG-based multimodal interface with application to humanoid robot control". In: *IEEE Transactions on Biomedical Engineering* 61.2 (2013), pp. 453–462.

[OA13] Alexis Ortiz-Rosario and Hojjat Adeli. "Brain-computer interface technologies: from signal to action". In: *Reviews in the Neurosciences* 24.5 (2013), pp. 537–552.

[Par+13] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. "Probabilistic Movement Primitives". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2013, pp. 2616–2624.

[AD14] Erwin Aertbeliën and Joris De Schutter. "eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 1540–1546.

[Man+14] Simon Manschitz, Jens Kober, Michael Gienger, and Jan Peters. "Learning to sequence movement primitives from demonstrations". In: *Int. Conf. Intelligent Robots and Systems*. IEEE. 2014, pp. 4414–4421.

[Iso+15] Mark Ison, Ivan Vujaklija, Bryan Whitsell, Dario Farina, and Panagiotis Artemiadis. "High-density electromyography and motor skill learning for robust long-term control of a 7-DoF robot arm". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering (TNSRE)* 24.4 (2015), pp. 424–433.

[Jai+15a] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. "Learning preferences for manipulation tasks from online coactive feedback". In: *The International Journal of Robotics Research* 34.10 (2015), pp. 1296–1313.

[Jai+15b] Siddarth Jain, Ali Farshchiansadegh, Alexander Broad, Farnaz Abdollahi, Ferdinando Mussa-Ivaldi, and Brenna Argall. "Assistive robotic manipulation through shared autonomy and a body-machine interface". In: *International Conference on Rehabilitation Robotics (ICORR)* (2015), pp. 526–531.

[Kam+15] Hyeong Ryeol Kam, Sung-Ho Lee, Taejung Park, and Chang-Hun Kim. "Rviz: a toolkit for real domain data visualization". In: *Telecommunication Systems* 60 (2015), pp. 337–345.

[KB15] Iosif Papadakis Ktistakis and Nikolaos G Bourbakis. “A survey on robotic wheelchairs mounted with robotic arms”. In: *2015 National Aerospace and Electronics Conference (NAECON)*. IEEE. 2015, pp. 258–262.

[Nie+15] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. “Learning grounded finite-state representations from unstructured demonstrations”. In: *International Journal of Robotics Research (IJRR)* 34.2 (2015), pp. 131–157.

[Arm+16] Brian S Armour, Elizabeth A Courtney-Long, Michael H Fox, Heidi Fredine, and Anthony Cahill. “Prevalence and causes of paralysis”. In: *American journal of public health (AJPH)* 106.10 (2016), pp. 1855–1857.

[BKM16] Siddharth Bhardwaj, Abid Ali Khan, and Mohammad Muzammil. “Electromyography in physical rehabilitation: a review”. In: *National Conference on Mechanical Engineering–Ideas, Innovations & Initiatives (NCMEI3)* (2016), pp. 64–69.

[BA16] Alexander Broad and Brenna Argall. “Path planning under interface-based constraints for assistive robotics”. In: *Twenty-Sixth International Conference on Automated Planning and Scheduling*. 2016.

[Bru+16] Sebastian G. Brunner, Franz Steinmetz, Rico Belder, and Andreas Dömel. “RAFCON: A graphical tool for engineering complex, robotic tasks”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), pp. 3283–3290.

[CAT16] Gerard Canal, Guillem Alenyà, and Carme Torras. “Personalization framework for adaptive robotic feeding assistance”. In: *Social Robotics: 8th International Conference, ICSR 2016, Kansas City, MO, USA, November 1-3, 2016 Proceedings 8*. Springer. 2016, pp. 22–31.

[Ewe+16] Marco Ewerton, Guilherme Maeda, Gerrit Kollegger, Josef Wiemeyer, and Jan Peters. “Incremental imitation learning of context-dependent motor skills”. In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2016, pp. 351–358.

[HHS16] Laura V Herlant, Rachel Holladay, and Siddhartha Srinivasa. “Assistive teleoperation of robot arms via automatic time-optimal mode switching”. In: *The Eleventh ACM/IEEE International Conference on Human Robot Interaction (HRI)* (2016), pp. 35–42.

[Jia+16] Hairong Jiang, Ting Zhang, Juan P Wachs, and Bradley S Duerstock. “Enhanced control of a wheelchair-mounted robotic manipulator using 3-D vision and multimodal interaction”. In: *Computer Vision and Image Understanding* 149 (2016), pp. 21–31.

[MHD16] Negar Mehr, Roberto Horowitz, and Anca D Dragan. “Inferring and assisting with constraints in shared autonomy”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)* (2016), pp. 6689–6696.

[Men+16] Jianjun Meng, Shuying Zhang, Angeliki Bekyo, Jaron Olsoe, Bryan Baxter, and Bin He. “Noninvasive electroencephalogram based control of a robotic arm for reach and grasp tasks”. In: *Scientific Reports* 6.1 (2016), pp. 1–15.

[Nie+16] Günter Niemeyer, Carsten Preusche, Stefano Stramigioli, and Dongjun Lee. In: *Springer handbook of robotics: Telerobotics*. Springer, 2016, pp. 1085–1108.

[Phi+16] Mike Phillips, Victor Hwang, Sachin Chitta, and Maxim Likhachev. “Learning to plan for constrained manipulation from demonstrations”. In: *Autonomous Robots* 40 (2016), pp. 109–124.

[Vog+16] Jörn Vogel, Katharina Hertkorn, Rohit U Menon, and Máximo A Roa. “Flexible, semi-autonomous grasping for assistive robotics”. In: *Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2016, pp. 4872–4879.

[WDV16] Roman Weitschat, Alexander Dietrich, and Jörn Vogel. "Online motion generation for mirroring human arm motion". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), pp. 4245–4250.

[Abi+17] Firas Abi-Farraj, Takayuki Osa, Nicoló Pedemonte Jan Peters, Gerhard Neumann, and Paolo Robuffo Giordano. "A learning-based shared control architecture for interactive task execution". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 329–335.

[Baj+17] Andrea Bajcsy, Dylan P Losey, Marcia K O'malley, and Anca D Dragan. "Learning robot objectives from physical human interaction". In: *Conference on robot learning*. PMLR. 2017, pp. 217–226.

[Bal+17] Tommaso Lisini Baldi, Giovanni Spagnoletti, Mihai Dragusanu, and Domenico Prattichizzo. "Design of a wearable interface for lightweight robotic arm for people with mobility impairments". In: *2017 International Conference on Rehabilitation Robotics (ICORR)*. IEEE. 2017, pp. 1567–1573.

[Bro+17] Alexander Broad, Jacob Arkin, Nathan Ratliff, Thomas Howard, and Brenna Argall. "Real-time natural language corrections for assistive robotic manipulators". In: *The International Journal of Robotics Research* 36.5-7 (2017), pp. 684–698.

[Chu+17] Cheng-Shiu Chung, Hyun W Ka, Hongu Wang, Dan Ding, Annmarie Kelleher, and Rory A Cooper. "Performance evaluation of a mobile touchscreen interface for assistive robotic manipulators: A pilot study". In: *Topics in spinal cord injury rehabilitation* 23.2 (2017), pp. 131–139.

[Mue+17] Katharina Muelling et al. "Autonomy infused teleoperation with application to brain computer interface controlled manipulation". In: *Autonomous Robots* 41.6 (2017), pp. 1401–1422.

[PS17] Claudia Pérez-D'Arpino and Julie Shah. "C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4058–4065.

[PC17] Emmanuel Pignat and Sylvain Calinon. "Learning adaptive dressing assistance from human demonstration". In: *Robotics and Autonomous Systems* 93 (2017), pp. 61–75.

[AC18] S Reza Ahmadzadeh and Sonia Chernova. "Trajectory-based skill learning using generalized cylinders". In: *Frontiers in Robotics and AI* 5 (2018), p. 132.

[AD18] Victoria Alonso and Paloma De La Puente. "System transparency in shared autonomy: A mini review". In: *Frontiers in neurorobotics* 12 (2018), p. 83.

[Baj+18] Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. "Learning from physical human corrections, one feature at a time". In: *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. 2018, pp. 141–149.

[Cam+18] Alexandre Campeau-Lecours, Ulysse Côté-Allard, Dinh-Son Vu, François Routhier, Benoit Gosselin, and Clément Gosselin. "Intuitive adaptive orientation control for enhanced human–robot interaction". In: *IEEE Transactions on robotics* 35.2 (2018), pp. 509–520.

[Fal+18] Cheikh Latyr Fall, Francis Quevillon, Martine Blouin, Simon Latour, Alexandre Campeau-Lecours, Clément Gosselin, and Benoit Gosselin. "A multimodal adaptive wireless control interface for people with upper-body disabilities". In: *IEEE transactions on biomedical circuits and systems* 12.3 (2018), pp. 564–575.

[HV18] Annette Hagengruber and Jörn Vogel. “Functional tasks performed by people with severe muscular atrophy using an sEMG controlled robotic manipulator”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2018), pp. 1713–1718.

[Jav+18] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha Srinivasa, and J Andrew Bagnell. “Shared autonomy via hindsight optimization for teleoperation and teaming”. In: *International Journal of Robotics Research (IJRR)* 37.7 (2018), pp. 717–742.

[KD18] Ayse Kucukyilmaz and Yiannis Demiris. “Learning shared control by demonstration for personalized wheelchair assistance”. In: *IEEE transactions on haptics* 11.3 (2018), pp. 431–442.

[Rai+18] Gennaro Raiola, Susana Sanchez Restrepo, Pauline Chevalier, Pedro Rodriguez-Ayerbe, Xavier Lamy, Sami Tliba, and Freek Stulp. “Co-manipulation with a Library of Virtual Guiding Fixtures”. In: *Autonomous Robots* (2018), pp. 1573–7527.

[Sel+18] Mario Selvaggio, Giuseppe Andrea Fontanelli, Fanny Ficuciello, Luigi Villani, and Bruno Siciliano. “Passive virtual fixtures adaptation in minimally invasive robotic surgery”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3129–3136.

[SZG18] Guru Subramani, Michael Zinn, and Michael Gleicher. “Inferring geometric constraints in human demonstrations”. In: *Conference on Robot Learning*. PMLR. 2018, pp. 223–236.

[VH18] Jörn Vogel and Annette Hagengruber. “An sEMG-based interface to give people with severe muscular atrophy control over assistive devices”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2018), pp. 2136–2141.

[WI18] Sebastian Wolf and Maged Iskandar. “Extending a dynamic friction model with nonlinear viscous and thermal dependency for a motor and harmonic drive gear”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 783–790.

[ZHC18] Martijn J A Zeestraten, Ioannis Havoutis, and Sylvain Calinon. “Programming by demonstration for shared control with an application in teleoperation”. In: *IEEE Robotics and Automation Letters (RA-L)* 3.3 (2018), pp. 1848–1855.

[Abi+19] Reza Abiri, Soheil Borhani, Eric W Sellers, Yang Jiang, and Xiaopeng Zhao. “A comprehensive review of EEG-based brain–computer interface paradigms”. In: *Journal of neural engineering (JNE)* 16.1 (2019), p. 011001.

[Cho+19] Ajit M Choudhari, Prasanna Porwal, Venkatesh Jonnalagedda, and Fabrice Mériaudeau. “An electrooculography based human machine interface for wheelchair control”. In: *Biocybernetics and Biomedical Engineering* 39.3 (2019), pp. 673–685.

[Dev+19] Louise Devigne, François Pasteau, Tom Carlson, and Marie Babel. “A shared control solution for safe assisted power wheelchair navigation in an environment consisting of negative obstacles: a proof of concept”. In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE. 2019, pp. 1043–1048.

[DO19] Alexander Dietrich and Christian Ott. “Hierarchical impedance-based tracking control of kinematically redundant robots”. In: *IEEE Transactions on Robotics* 36.1 (2019), pp. 204–221.

[Gal+19] Daniel Gallenberger, Tapomayukh Bhattacharjee, Youngsun Kim, and Siddhartha S Srinivasa. “Transfer depends on acquisition: Analyzing manipulation strategies for robotic feeding”. In: *Int. Conf. Human-Robot Interaction (HRI)*. IEEE. 2019, pp. 267–276.

[GK19] Phillip M Grice and Charles C Kemp. “In-home and remote use of robotic body surrogates by people with profound motor deficits”. In: *PloS one* 14.3 (2019), e0212904.

[Hua+19] Yanlong Huang, Leonel Rozo, Joao Silvério, and Darwin G Caldwell. “Kernelized movement primitives”. In: *The International Journal of Robotics Research* 38.7 (2019), pp. 833–852.

[Isk+19] Maged Iskandar, Gabriel Quere, Annette Hagengruber, Alexander Dietrich, and Jörn Vogel. “Employing whole-body control in assistive robotics”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 5643–5650.

[IW19] Maged Iskandar and Sebastian Wolf. “Dynamic friction model with thermal and load dependency: modeling, compensation, and external force estimation”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7367–7373.

[JA19] Siddarth Jain and Brenna Argall. “Probabilistic human intent recognition for shared autonomy in assistive robotics”. In: *ACM Transactions on Human-Robot Interaction (THRI)* 9.1 (2019), pp. 1–23.

[Lei19] Daniel Sebastian Leidner. *Cognitive reasoning for compliant robot manipulation*. Springer, 2019.

[Moh+19] Anahita Mohseni-Kabir et al. “Simultaneous learning of hierarchy and primitives for complex robot tasks”. In: *Autonomous Robots* 43 (2019), pp. 859–874.

[ORR19] Juan F Orejuela-Zapata, Sarita Rodriguez, and Gonzalo Llano Ramirez. “Self-help devices for quadriplegic population: A systematic literature review”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27.4 (2019), pp. 692–701.

[Rah+19] Rahaf Rahal, Firas Abi-Farraj, Paolo Robuffo Giordano, and Claudio Pacchierotti. “Haptic shared-control methods for robotic cutting under nonholonomic constraints”. In: *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2019, pp. 8151–8157.

[SOF19] Ali Shafti, Pavel Orlov, and A Aldo Faisal. “Gaze-based, context-aware robotic system for assisted reaching and grasping”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 863–869.

[Bal+20] Ribin Balachandran, Hrishik Mishra, Matteo Cappelli, Bernhard Weber, Cristian Secchi, Christian Ott, and Alin Albu-Schäffer. “Adaptive authority allocation in shared control of robots using Bayesian filters”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 11298–11304.

[Bha+20] Tapomayukh Bhattacharjee, Ethan K Gordon, Rosario Scalise, Maria E Cabrera, Anat Caspi, Maya Cakmak, and Siddhartha S Srinivasa. “Is more autonomy always better? exploring preferences of users with mobility impairments in robot-assisted feeding”. In: *Proceedings of the 2020 ACM/IEEE international conference on human-robot interaction*. 2020, pp. 181–190.

[Eri+20] Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C Karen Liu, and Charles C Kemp. “Assistive gym: A physics simulation framework for assistive robotics”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 10169–10176.

[FR20] Werner Friedl and Máximo A Roa. “CLASH—A Compliant Sensorized Hand for Handling Delicate Objects”. In: *Frontiers in Robotics and AI* 6 (2020), p. 138.

[GA20] Deepak E Gopinath and Brenna D Argall. “Active intent disambiguation for shared control robots”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering (TNSRE)* 28.6 (2020), pp. 1497–1506.

[Isk+20] Maged Iskandar, Christian Ott, Oliver Eiberger, Manuel Keppler, Alin Albu-Schäffer, and Alexander Dietrich. "Joint-level control of the DLR lightweight robot SARA". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), pp. 8903–8910.

[JA20] Mahdieh Nejati Javaremi and Brenna D Argall. "Characterization of assistive robot arm teleoperation: A preliminary study to inform shared control". In: *arXiv preprint, arXiv:2008.00109* (2020).

[Los+20] Dylan P Losey, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, and Dorsa Sadigh. "Controlling assistive robots with learned latent actions". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 378–384.

[Par+20] Daehyung Park, Yuuna Hoshi, Harshal P Mahajan, Ho Keun Kim, Zackory Erickson, Wendy A Rogers, and Charles C Kemp. "Active robot-assisted feeding with a general-purpose mobile manipulator: Design, evaluation, and lessons learned". In: *Robotics and Autonomous Systems* 124 (2020), p. 103344.

[Que+20] Gabriel Quere, Annette Hagengruber, Maged Iskandar, Samuel Bustamante, Daniel Leidner, Freek Stulp, and Jörn Vogel. "Shared control templates for assistive robotics". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 1956–1962.

[Ras+20] Mamunur Rashid, Norizam Sulaiman, Anwar PP Abdul Majeed, Rabiu Muazu Musa, Fakhri Ab. Nasir, Bifta Sama Bari, and Sabira Khatun. "Current status, challenges, and possible solutions of EEG-based brain-computer interface: a comprehensive review". In: *Frontiers in neurorobotics* (2020), p. 25.

[Vog+20a] Jorn Vogel et al. "An ecosystem for heterogeneous robotic assistants in caregiving: Core functionalities and use cases". In: *IEEE Robotics & Automation Magazine* 28.3 (2020), pp. 12–28.

[Vog+20b] Jörn Vogel et al. "Edan: An emg-controlled daily assistant to help people with physical disabilities". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 4183–4190.

[WEL20] Christoph Willibald, Thomas Eiband, and Dongheui Lee. "Collaborative Programming of Conditional Robot Tasks". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5402–5409.

[ZZZ20] Zhenliang Zhang, Yixin Zhu, and Song-Chun Zhu. "Graph-based hierarchical knowledge representation for robot task transfer from virtual to physical world". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 11139–11145.

[Bus+21] Samuel Bustamante, Gabriel Quere, Katharina Hagmann, Xuwei Wu, Peter Schmaus, Jörn Vogel, Freek Stulp, and Daniel Leidner. "Toward seamless transitions between shared control and supervised autonomy in robotic assistance". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3833–3840.

[CTA21] Gerard Canal, Carme Torras, and Guillem Alenyà. "Are Preferences Useful for Better Assistance?: A Physically Assistive Robotics User Study". In: *ACM Transactions on Human-Robot Interaction (THRI)* 10.4 (2021), pp. 1–19.

[CBO21] Glen Chou, Dmitry Berenson, and Necmiye Ozay. "Learning constraints from demonstrations with grid and parametric representations". In: *The International Journal of Robotics Research* 40.10-11 (2021), pp. 1255–1283.

[Die+21] Alexander Dietrich, Xuwei Wu, Kristin Bussmann, Marie Harder, Maged Iskandar, Johannes Englsberger, Christian Ott, and Alin Albu-Schäffer. "Practical consequences of inertia shaping for interaction and tracking in robot control". In: *Control Engineering Practice* 114 (2021), p. 104875.

[Fle+21] Aaron Fleming, Nicole Stafford, Stephanie Huang, Xiaogang Hu, Daniel P Ferris, and He Helen Huang. "Myoelectric control of robotic lower limb prostheses: a review of electromyography interfaces, control paradigms, challenges and future directions". In: *Journal of neural engineering (JNE)* 18.4 (2021), p. 041004.

[Hag+21] Annette Hagengruber, Ulrike Leipscher, Bjoern M Eskofier, and Jörn Vogel. "Electromyography for Teleoperated Tasks in Weightlessness". In: *IEEE Transactions on Human-Machine Systems (T-HMS)* 51.2 (2021), pp. 130–140.

[IDA21] Santiago Iregui, Joris De Schutter, and Erwin Aertbeliën. "Reconfigurable Constraint-Based Reactive Framework for Assistive Robotics With Adaptable Levels of Autonomy". In: *IEEE Robotics and Automation Letters (RA-L)* 6.4 (2021), pp. 7397–7405.

[Isk+21] Maged Iskandar, Oliver Eiberger, Alin Albu-Schäffer, Alessandro De Luca, and Alexander Dietrich. "Collision detection, identification, and localization on the DLR SARA robot with sensing redundancy". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), pp. 3111–3117.

[KNK21] Oliver Kroemer, Scott Niekum, and George Konidaris. "A review of robot learning for manipulation: Challenges, representations, and algorithms". In: *Journal of machine learning research* 22.30 (2021), pp. 1–82.

[Mic+21] Youssef Michel, Rahaf Rahal, Claudio Pacchierotti, Paolo Robuffo Giordano, and Dongheui Lee. "Bilateral teleoperation with adaptive impedance control for contact tasks". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5429–5436.

[MZR21] Sina Mohseni, Niloofar Zarei, and Eric D Ragan. "A multidisciplinary survey and framework for design and evaluation of explainable AI systems". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 11.3-4 (2021), pp. 1–45.

[Qia+21] Calvin Z Qiao, Maram Sakr, Katharina Muelling, and Henny Admoni. "Learning from demonstration for real-time user goal prediction and shared assistive control". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 3270–3275.

[Que+21] Gabriel Quere, Samuel Bustamante, Annette Hagengruber, Jörn Vogel, Franz Steinmetz, and Freek Stulp. "Learning and Interactive Design of Shared Control Templates". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), pp. 1887–1894.

[Bel+22] Suneel Belkhale, Ethan K. Gordon, Yuxiao Chen, Siddhartha Srinivasa, Tapomayukh Bhattacharjee, and Dorsa Sadigh. "Balancing Efficiency and Comfort in Robot-Assisted Bite Transfer". In: *2022 IEEE International Conference on Robotics and Automation (ICRA)* (2022).

[Bus+22] Samuel Bustamante, Gabriel Quere, Daniel Leidner, Jörn Vogel, and Freek Stulp. "CATs: Task Planning for Shared Control of Assistive Robots with Variable Autonomy". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 3775–3782.

[Cel+22] Carlos Celemin et al. "Interactive imitation learning in robotics: A survey". In: *Foundations and Trends® in Robotics* 10.1-2 (2022), pp. 1–197.

[Got+22] Alberto Gottardi, Stefano Tortora, Elisa Tosello, and Emanuele Menegatti. "Shared Control in Robot Teleoperation With Improved Potential Fields". In: *IEEE Transactions on Human-Machine Systems (T-HMS)* 52.3 (2022), pp. 410–422.

[How+22] Jonathan Howard, Zoe Fisher, Andrew H Kemp, Stephen Lindsay, Lorna H Tasker, and Jeremy J Tree. "Exploring the barriers to using assistive technology for individuals with chronic conditions: a meta-synthesis review". In: *Disability and rehabilitation: Assistive technology* 17.4 (2022), pp. 390–408.

[Isk+22] Maged Iskandar, Christiaan van Ommeren, Xuwei Wu, Alin Albu-Schäffer, and Alexander Dietrich. "Model predictive control applied to different time-scale dynamics of flexible joint robots". In: *IEEE Robotics and Automation Letters* 8.2 (2022), pp. 672–679.

[Los+22] Dylan P Losey, Hong Jun Jeon, Mengxi Li, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, Jeannette Bohg, and Dorsa Sadigh. "Learning latent actions to control assistive robots". In: *Autonomous robots* 46.1 (2022), pp. 115–147.

[Lut22] Matthias Lutz. "Composable Coordination for Service Robots: A Model-Driven Approach". PhD thesis. Technische Universität München, 2022.

[NH22] Patrick Naughton and Kris Hauser. "Structured Action Prediction for Teleoperation in Open Worlds". In: *IEEE Robotics and Automation Letters (RA-L)* 7.2 (2022), pp. 3099–3105.

[Pet+22] Laura Petrich, Jun Jin, Masood Dehghan, and Martin Jagersand. "A quantitative analysis of activities of daily living: Insights into improving functional independence with assistive robotics". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 6999–7006.

[WL22] Christoph Willibald and Dongheui Lee. "Multi-level task learning based on intention and constraint inference for autonomous robotic manipulation". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 7688–7695.

[Ye+22] Ruolin Ye, Wenqiang Xu, Haoyuan Fu, Rajat Kumar Jenamani, Vy Nguyen, Cewu Lu, Katherine Dimitropoulou, and Tapomayukh Bhattacharjee. "Rcare world: A human-centric simulation world for caregiving robots". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 33–40.

[Den+23] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Wendelin Knauer, Klaus H Strobl, Matthias Humt, and Rudolph Triebel. "Blenderproc2: A procedural pipeline for photorealistic rendering". In: *Journal of Open Source Software* 8.82 (2023), p. 4901.

[Gor+23] Ethan Kroll Gordon, Amal Nanavati, Ramya Challa, Bernie Hao Zhu, Taylor Annette Kessler Faulkner, and Siddhartha Srinivasa. "Towards general single-utensil food acquisition with human-informed actions". In: *Conference on Robot Learning*. PMLR. 2023, pp. 2414–2428.

[Isk+23] Maged Iskandar, Christian Ott, Alin Albu-Schäffer, Bruno Siciliano, and Alexander Dietrich. "Hybrid force-impedance control for fast end-effector motions". In: *IEEE Robotics and Automation Letters* 8.7 (2023), pp. 3931–3938.

[Jae+23] Lukas Jaeger et al. "How the CYBATHLON competition has advanced assistive technologies". In: *Annual Review of Control, Robotics, and Autonomous Systems* 6.1 (2023), pp. 447–476.

[Pad+23] Abhishek Padalkar, Gabriel Quere, Franz Steinmetz, Antonin Raffin, Matthias Nieuwenhuisen, João Silvério, and Freek Stulp. "Guiding Reinforcement Learning with Shared Control Templates". In: *40th IEEE International Conference on Robotics and Automation, ICRA 2023*. IEEE. 2023.

[Prz+23] Michael Przystupa, Kerrick Johnstonbaugh, Zichen Zhang, Laura Petrich, Masood Dehghan, Faezeh Haghverd, and Martin Jagersand. "Learning State Conditioned Linear Mappings for Low-Dimensional Control of Robotic Manipulators". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 857–863.

[She+23] Afagh Mehri Shervedani, Siyu Li, Natawut Monaikul, Bahareh Abbasi, Barbara Di Eugenio, and Miloš Žefran. "An end-to-end human simulator for task-oriented multimodal human-robot collaboration". In: *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE. 2023, pp. 614–620.

[SH23] João Silvério and Yanlong Huang. "A Non-parametric Skill Representation with Soft Null Space Projectors for Fast Generalization". In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. 2023, pp. 2988–2994.

[Sto+23] Manuel Stoiber, Mariam Elsayed, Anne E Reichert, Florian Steidle, Dongheui Lee, and Rudolph Triebel. "Fusing Visual Appearance and Geometry for Multi-modality 6DoF Object Tracking". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 1170–1177.

[Ulm+23] Maximilian Ulmer, Maximilian Durner, Martin Sundermeyer, Manuel Stoiber, and Rudolph Triebel. "6d object pose estimation from approximate 3d models for orbital robotics". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 10749–10756.

[WBL23] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 7464–7475.

[Bus+24] Samuel Bustamante et al. "Feasibility Checking and Constraint Refinement for Shared Control in Assistive Robotics". In: *IEEE Robotics and Automation Letters* (2024).

[Hag+24] Katharina Hagmann, Anja Hellings-Kuss, Florian Steidle, Freek Stulp, Daniel Leidner, and Julian Klodmann. "Continuous Transitions between Levels of Autonomy based on Virtual Fixtures for Surgical Robotic Systems". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2024*. IEEE. 2024.

[IAD24] Maged Iskandar, Alin Albu-Schäffer, and Alexander Dietrich. "Intrinsic sense of touch for intuitive physical human-robot interaction". In: *Science Robotics* 9.93 (2024), eadn4008.

[Mil+24] Elle Miller, Maximilian Durner, Matthias Humt, Gabriel Quere, Wout Boerdijk, Ashok M Sundaram, Freek Stulp, and Jörn Vogel. "Unknown object grasping for assistive robotics". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 18157–18163.

[Pad+24] Akhil Padmanabha, Janavi Gupta, Chen Chen, Jehan Yang, Vy Nguyen, Douglas J Weber, Carmel Majidi, and Zackory Erickson. "Independence in the Home: A Wearable Interface for a Person with Quadriplegia to Teleoperate a Mobile Manipulator". In: *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 2024, pp. 542–551.

[Pas+24] Max Pascher, Felix Ferdinand Goldau, Kirill Kronhardt, Udo Frese, and Jens Gerken. "AdaptiX-A Transitional XR Framework for Development and Evaluation of Shared Control Applications in Assistive Robotics". In: *Proceedings of the ACM on Human-Computer Interaction* 8.EICS (2024), pp. 1–28.

[Que+24] Gabriel Quere, Freek Stulp, David Filliat, and João Silvério. "A probabilistic approach for learning and adapting shared control skills with the human in the loop". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)* (2024).

[Sch+24] Philipp Scholl, Maged Iskandar, Sebastian Wolf, Jinoh Lee, Aras Bacho, Alexander Dietrich, Alin Albu-Schäffer, and Gitta Kutyniok. “Learning-based adaption of robotic friction models”. In: *Robotics and Computer-Integrated Manufacturing* 89 (2024), p. 102780.

[Hag+25] Annette Hagengruber, Gabriel Quere, Samuel Bustamante, Jianxiang Feng, Daniel Leidner, Alin Albu-Schäffer, Freek Stulp, and Jörn Vogel. “An assistive robot that enables people with amyotrophia to perform sequences of everyday activities”. In: *Scientific reports* (2025).

[Org] World Health Organization. *Spinal cord injury*. `https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury`. Accessed: 2024-09-23.

[Zür] CYBATHLON ETH Zürich. *Cybathlon*. `https://cybathlon.ethz.ch/en/cybathlon`. Accessed: 2024-09-23.

Chapter 9

Appendix

A few works have made use of the Shared Control Templates framework.

9.1 Toward Seamless Transitions Between Shared Control and Supervised Autonomy in Robotic Assistance

Abstract by Bustamante *et al.* [Bus+21]:

"Assistive robots aim to help humans with impairments execute motor tasks in everyday household environments. Controlling the end-effector of such robots directly, for instance with a joystick, is often cumbersome. Shared control methods, like Shared Control Templates (SCTs) , have therefore been proposed to provide support for robotic control. Moreover, depending on factors such as workload, system trust or engagement, users may like to freely adjust the level of autonomy, for instance by letting the robot complete a task by itself. In this letter, we present a concept for adjustable autonomy in the context of robotic assistance. We extend the SCT approach with an automatic control module that allows the user to switch between Shared Control and Supervised Autonomy at any time during task execution. As both support modes use the same action representation, transitions are seamless. We show the capabilities of this approach in a set of daily living tasks with our wheelchair-mounted robot EDAN and our humanoid robot Rollin' Justin. We highlight how automatic execution benefits from SCT features, like task-related constraints and whole-body control."

9.2 CATs: Task Planning for Shared Control of Assistive Robots with Variable Autonomy

Abstract by Bustamante *et al.* [Bus+22]:

"From robotic space assistance to healthcare robotics, there is increasing interest in robots that offer adaptable levels of autonomy. In this paper, we propose an action representation and planning framework that is able to generate plans that can be executed with both shared control and supervised autonomy, even switching between them during task execution. The action representation - Constraint Action Templates (CATs) - combine the advantages of Action Templates and Shared Control Templates. We demonstrate that CATs enable our planning framework to generate goal-directed plans for variations of a typical task of daily living, and that users can execute them on the wheelchair-robot EDAN in shared control or in autonomous mode."

9.3 Guiding Reinforcement Learning with Shared Control Templates

Abstract by Padalkar *et al.* [Pad+23]:

"Purposeful interaction with objects usually requires certain constraints to be respected, e.g. keeping a bottle upright to avoid spilling. In reinforcement learning, such constraints are typically encoded in the reward function. As a consequence, constraints can only be learned by violating them. This often precludes learning on the physical robot, as it may take many trials to learn the constraints, and the necessity to violate them during the trial-and-error learning may be unsafe. We have serendipitously discovered that constraint representations for shared control – in particular Shared Control Templates (SCTs) – are ideally suited for guiding RL. Representing constraints explicitly (rather than implicitly in the reward function) also simplifies the design of the reward function. We evaluate the advantages of the approach (faster learning without constraint violations, even with sparse reward functions) in a simulated pouring task. Furthermore, we demonstrate that these advantages enable the real robot to learn this task in only 65 episodes taking 16 minutes."

9.4 Unknown Object Grasping for Assistive Robotics

Abstract by Miller *et al.* [Mil+24]:

"We propose a novel pipeline for unknown object grasping in shared robotic autonomy scenarios. State-of-the-art methods for fully autonomous scenarios are typically learning-based approaches optimised for a specific end-effector, that generate grasp poses directly from sensor input. In the domain of assistive robotics, we seek instead to utilise the user's cognitive abilities for enhanced satisfaction, grasping performance, and alignment with their high level task-specific goals. Given a pair of stereo images, we perform unknown object instance segmentation and generate a 3D reconstruction of the object of interest. In shared control, the user then guides the robot end-effector across a virtual hemisphere centered around the object to their desired approach direction. A physics-based grasp planner finds the most stable local grasp on the reconstruction, and finally the user is guided by shared control to this grasp. In experiments on the DLR EDAN platform, we report a grasp success rate of 87% for 10 unknown objects, and demonstrate the method's capability to grasp objects in structured clutter and from shelves."

9.5 Continuous Transitions between Levels of Autonomy based on Virtual Fixtures for Surgical Robotic Systems

Abstract by Hagmann *et al.* [Hag+24]:

"Nowadays, telemanipulation robotic systems are present in many operating rooms. The exploitation of autonomy for minimally invasive robotic surgery remains an open field of research as it is non-trivial to provide meaningful assistance. This work presents a novel virtual fixture providing haptic augmentation for shared control as well as task-level autonomy, while ensuring continuous transitions of control between the robotic system and the surgeon. Transitions between levels of autonomy are based on information about the robotic system and its environment. The proposed method is evaluated through experiments which show the successful completion of surgeon training tasks, namely peg transfer and suturing, exploiting shared control and task-level autonomy."

ECOLE DOCTORALE

Titre : Apprentissage et conception de compétences de contrôle partagé pour robots d'assistance

Mots clés : Contrôle partagé, Robots d'assistance, Apprentissage de compétence

Résumé :

Les robots d'assistance, tels qu'un bras robotique monté sur un fauteuil roulant, offrent une aide précieuse aux personnes handicapées moteurs en les aidant à effectuer des tâches quotidiennes comme manger ou ouvrir une porte. Cependant, l'utilisation de ces robots peut s'avérer difficile en raison de leur complexité. Cette thèse présente une nouvelle approche qui aide l'utilisateur pour ses tâches tout en lui permettant de garder le contrôle sur les actions clés - comme décider de la quantité d'eau à verser d'un thermos alors que le robot évite les déversements - assurant une plus grande fiabilité dans l'exécution de ces tâches. De nouvelles compétences peuvent être acquises à partir de l'enregistrement de trajectoires du robot pour de nouvelles tâches. Des expériences menées avec notre robot d'assistance, EDAN, ont montré que des utilisateurs valides et handicapés étaient en mesure d'accomplir avec succès des activités de la vie quotidienne.

Title : Learning and designing shared control skills from demonstrations for assistive robots

Keywords : Shared control, Assitive robots, Skill learning

Abstract :

Assistive robots, such as wheelchair-mounted robotic arms, offer valuable support to individuals with limited physical capabilities by helping them perform everyday tasks like eating or navigating through doors. However, operating these robots using interfaces such as joysticks can be difficult due to the system complexity and variety of possible movements. This thesis presents a new approach that provides task-specific assistance while keeping the user in control over key actions - such as deciding how much water to pour from a thermos while the robot ensures no spilling - providing greater reliability in task execution and enhancing users' abilities to interact with their environment. New skills can be learned from recorded task executions and users can adapt skills to new conditions. Experiments using our assistive robot, EDAN, showed that able-bodied and motor-impaired users were able to successfully perform daily life activities.

Institut Polytechnique de Paris
91120 Palaiseau, France