# Cognitive shared control in assistive robotics

**By Samuel Bustamante Gomez, M.Sc.**

This is a preliminary version of this doctoral thesis. This is the version of the manuscript submitted in December 2025 for consideration at the Technical University of Munich. The final version in accordance with the changes suggested by the examination committee will be uploaded once it is available.

## Abstract

Input devices for assistive robots, such as a joystick or an electromyography sensor array on the skin's surface, often have low-dimensional input spaces. While using such a low-dimensional input device to control an assistive robot with many degrees of freedom can be challenging and frustrating, users with motor impairments express satisfaction with having control authority over the robot's movements, even if this leads to higher workloads. In this context, so-called shared control frameworks can provide support to users while empowering them to be in control of the task.

This thesis explores how knowledge-based robots that interpret and understand their environment can support users on various levels of abstraction. More concretely, I show how endowing shared control frameworks with the ability to represent, acquire, and process knowledge — in other words, by realizing **cognitive shared control** — can provide users with four desired features: **flexibility** on the amount of control they can have; **robustness** during shared control tasks; **guidance** towards a specific **goal**; and **explainability** of the robotic behavior. I further show how these features have been achieved by presenting a set of methods, as well as experiments conducted on the wheelchair-based mobile manipulator EDAN and the humanoid assistant Rollin' Justin, two robots developed by the German Aerospace Center (DLR).

## Zusammenfassung

Eingabegeräte für Assistenzroboter, wie etwa ein Joystick oder ein elektromyographisches Sensorarray auf der Hautoberfläche, haben oft einen niedrigdimensionalen Eingabebereich. Bei der Steuerung eines Hilfsroboters mit vielen Freiheitsgraden kann die Verwendung eines solchen niedrigdimensionalen Eingabegeräts schwierig und frustrierend sein. Jedoch äußern Benutzer mit motorischen Beeinträchtigungen oft Zufriedenheit darüber, die Kontrolle über die Bewegungen des Roboters zu haben, selbst dann, wenn dies zu einer höheren Arbeitsbelastung führt. In diesem Kontext können Methoden der sogenannten geteilten Steuerung nützlich sein, um die Benutzer zu unterstützen und ihnen gleichzeitig die Kontrolle über die Aufgabe zu ermöglichen.

In dieser Arbeit wird untersucht, wie wissensbasierte Roboter, die die Umgebung interpretieren und verstehen, den Benutzern auf verschiedenen Abstraktionsebenen Unter-

stützung bieten können. Konkret zeige ich, wie die Erweiterung von geteilter Steuerung durch die Fähigkeit, Wissen zu repräsentieren, zu erwerben und zu verarbeiten — mit anderen Worten, **kognitionsbasierte geteilte Steuerung** — den Benutzern vier gewünschte Eigenschaften bringt: **Flexibilität** im Bezug auf die gewünschte Kontrollstufe; **Robustheit** während gemeinsamer Steuerungsaufgaben; **Lenkung**, um ein bestimmtes **Ziels** zu erreichen; und **Erklärbarkeit** des Roboterverhaltens. Ich zeige wie diese Eigenschaften erreicht wurden, anhand von Methoden sowie Experimenten mit dem rollstuhlbasierten mobilen Manipulator EDAN und dem humanoiden Assistenzroboter Rollin' Justin, zwei Robotern des Deutschen Zentrums für Luft- und Raumfahrt (DLR).

# Preface

## Usage of the pronouns "I" and "We"

As the reader will notice, I prefer using an active voice in the present text. I use the pronoun "I" whenever I write content that I have created for my thesis. I use the pronoun "we" to signal "I and my colleagues" whenever a procedure or result comes from joint work (e.g., a paper) with co-authors.

My thesis is based upon scientific work conducted between 2019 and 2025, particularly a main set of work (Section 1.6.1) from which I am the first author.

**Copyright notice:** All of my papers as first author were published in IEEE journals and conferences. As I add textual sections and figures (and cite them accordingly), I also add a copyright notice when the material's copyright belongs to IEEE. I quote:

> "The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant". Source: Frequently Asked Questions Regarding IEEE Permissions [87].

IEEE requires me to present the following notice:

> "In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of TUM or DLR's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. ".

## Usage of generative AI

I used generative AI in the present work as a writing tool to suggest rephrasings of text blocks, as well as to assist with repetitive tasks such as generating a bibtex file from a citation in a different format. I also utilized TUM's Grammarly service, as well as autoregressive language models. All textual generations were read, understood, and curated by me in accordance with the artificial intelligence strategy of the Technical University of Munich [134].

### Artist depictions

A set of five illustration panels depicting a user of an assistive robot was commissioned for the thesis and was drawn by Adelina Flórez. These panels are shown in figs. 1.3 and 1.9 (and small design elements were used on Fig. 7.1). The content of the illustrations was an explicit request of the thesis author. Adelina Flórez retains the moral rights to the illustrations, and the thesis author retains the copyright for them.

## Acknowledgements

This thesis encompasses several years of work at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR), where I appreciate the scientific environment. One is surrounded by many specialists who provide helpful hands, and learns something new daily.

I would like to thank Prof. Dr. Alin Albu-Schäffer for the opportunity to conduct research at DLR and for the patient supervision of this thesis. I extend my thanks to Dr. Freek Stulp, the head of my department at DLR and my "Fachbetreuer" (specialist advisor) during most of my doctorate years. Freek always gave wings to my ideas, and I am particularly thankful for his encouraging support during the most trying times of academic life, such as paper deadlines and rebuttals. I also thank Prof. Dr. Daniel Leidner, who was my first "Fachbetreuer" in his time as group leader, and helped me gain confidence in the research efforts that led to my first paper; Jörn Vogel and Annette Hagengruber, the leaders of the Reenabling Robotics team, who gave me enormous support and freedom through the years to bring my ideas into the EDAN system; and Peter Schmaus, who leads the Rollin' Justin team.

I am grateful to the many colleagues at DLR whose contributions have enriched this work. First and foremost, I would like to acknowledge Gabriel Quere, who worked closely with me during these years and always made time to discuss technical questions (and geopolitics) in our shared office; and Ismael Rodríguez, for conveying his expertise on feasibility checks, which contributed to the work in Chapter 4. I also thank the members of the Reenabling Robotics and Rollin' Justin teams, namely Maged Iskandar, Miriam Welser, Felix Schiel, Adrian Bauer, Xuwei Wu, Jianxiang Feng, Luisa Mayershofer, Nesrine Batti, Xiaozhou Luo, Alina Ostrowski, Max Denninger, Armand Garrigou, Lioba Schürmann, Simon Spielmann, and Anne Köpken; fellow hackathon collaborators Markus Knauer, Stefan Schneyer, Sebastian Jung, Leonard Klüpfel, Antonin Raffin, Maximilian Mühlbauer, Mathilde Conan, Andrea Gigli, Ryo Sakagami, Ashok Meenakshi Sundaram, Andrea Sipos and Florian Lay; supporting colleagues Klaus Strobl, Dominik Winkelbauer, Neal Y. Lii, Andreas Döml, João Silvério, Antonin Raffin, Xuming Meng, Ana Huezo, Peter Lehner, Robert Schuller, Jinoh Lee, Max Durner, Florian Schmidt, and George Mesesan; my former students Mohit Kumar, Jeremias Thun, Britt Besch, Valentina Choquenaira, Daniel Barros, and Juan Diego Plaza; external collaborators Santiago Irgeui (KU Leuven) and Daniel Beßler (Universität Bremen); the service team, particularly Ariane Schlinzig and Michaela Risch; and the IT team. Finally, I extend my thanks to the members of my topic group, Philipp Knestel, Bernhard Weber, and Tai Mai, for this refreshing new chapter in our research.

Thanks also to Adelina Flórez for her support and her illustrations for the introduction. Finally, I thank my family and friends for their support over the years, particularly to Milena, Paula, Gloria, Javier, Brian, Isa, Kathy, Khaled, Marlly, Pablo, David, and Felipe.

Thanks to all the people on this list, these years have been not only a considerable effort, but also very rewarding and great fun.

# Contents


**1 Introduction** **1**

1.1 Motivation . . . 1

1.2 Cognitive shared control and its desired features . . . 3

1.2.1 Contribution 1: shared control should be **flexible** . . . 4

1.2.2 Contribution 2: shared control should be **robust** . . . 6

1.2.3 Contribution 3: shared control should provide **guidance** towards a **goal** 8

1.2.4 Contribution 4: shared control should be **explainable** . . . 9

1.2.5 Summarizing example . . . 11

1.3 Knowledge representation for cognitive shared control . . . 11

1.4 Related work . . . 13

1.5 Summary: statement of contributions . . . 19

1.6 Thesis outline . . . 20

1.6.1 Publications note . . . 21

**2 Fundamentals** **25**

2.1 A taxonomy of shared control support in assistive robotics . . . 25

2.1.1 Branch 1: command support . . . 27

2.1.2 Branch 2: constraint support . . . 30

2.2 Knowledge representation with world models . . . 34

2.2.1 Concepts . . . 34

2.2.2 Implementation . . . 36

2.3 Shared Control Templates . . . 38

2.3.1 SCT input . . . 38

2.3.2 Finite State Machines (FSMs) . . . 39

2.3.3 Support within a state: Input Mappings and Active Constraints . . . 39

2.3.4 SCT output . . . 40

2.3.5 SCT activation via the world model and action graphs . . . 41

2.3.6 Ease of design: learning SCTs . . . 43

2.3.7 Limitations of the SCTs . . . 43

2.4 Robots . . . 44

2.5 Outlook of the contributions given the taxonomy . . . 45

**3 Shared control with integrated autonomy** **47**
3.1 Assumptions . . . 48
3.2 Shared Control with Integrated Autonomy . . . 49
3.2.1 Autonomous execution within a state . . . 50
3.2.2 Task completion . . . 57
3.2.3 Example tasks and robots . . . 59
3.3 Experiments . . . 60
3.3.1 Description: . . . 60
3.3.2 Experiment set 1: speed test . . . 61
3.3.3 Experiment set 2: activities of the daily living . . . 62
3.4 Discussion and outlook . . . 64

**4 Feasibility checks before shared control tasks** **67**
4.1 Preliminary: Task Space Regions for shared control . . . 68
4.1.1 Background: Task Space Regions . . . 69
4.1.2 Representing Active Constraints in SCTs with TSRs . . . 70
4.1.3 Refined TSRs . . . 72
4.1.4 Instantiating an SCT from a refined TSR . . . 72
4.2 Feasibility checking and constraint refinement for shared control . . . 73
4.2.1 Feasibility Checking . . . 74
4.2.2 Constraint refinement and SCT instantiation . . . 78
4.2.3 Motion planning for enabling infeasible tasks . . . 80
4.3 Experimental evaluation . . . 81
4.3.1 Appropriate SCT activation (simulation) . . . 83
4.3.2 Appropriate SCT activation (real robot) . . . 84
4.3.3 Computation times benchmark . . . 85
4.3.4 Discussion of the experimental results . . . 86
4.4 Limitations and outlook . . . 87
4.4.1 Faster feasibility checks . . . 87
4.4.2 Global feasibility checks . . . 87
4.4.3 Dynamic obstacles . . . 88

**5 Task planning for shared control with Constraint-Action Templates** **89**
5.1 Background: the Planning Domain Description Language . . . 90
5.1.1 Symbolic planning with PDDL . . . 90
5.1.2 Action Templates . . . 93
5.2 The Constraint Action Templates (CATs) . . . 95
5.2.1 Rationale and requirements . . . 95
5.2.2 Quick overview: . . . 97
5.2.3 Example and components of a CAT file . . . 97

5.2.4 Object class hierarchy: grounding of objects and reference frames . . . 99
5.2.5 Representing parameters for feasibility checks . . . 102
5.3 Planning and execution with CATs . . . 102
5.3.1 Symbolic planner . . . 102
5.3.2 Shared control planner . . . 102
5.3.3 Execution of the generated SCT . . . 105
5.4 Experiments . . . 105
5.4.1 Experiment 1: CAT plans . . . 105
5.4.2 Experiment 2: planning time duration . . . 106
5.5 Discussion, limitations and future works . . . 107
5.5.1 Extension concept: feasibility checks of complete task plans . . . 109

**6 Query-answering for robots with pre-existing modules 111**
6.1 Background: RAG . . . 112
6.2 RACCOON . . . 113
6.2.1 Assumptions . . . 114
6.2.2 Selecting the relevant robot module(s) from user queries . . . 114
6.2.3 Queries related to multiple classes: . . . 115
6.2.4 Retrieving state summaries from robot modules . . . 116
6.2.5 Generation . . . 117
6.2.6 Summary . . . 117
6.3 Examples of EDAN state summaries . . . 117
6.4 Architecture of EDAN integration: an explainability system based on speech . 120
6.4.1 input & output connection to the robot . . . 121
6.5 Experimental evaluation . . . 122
6.5.1 Experiment 1: truthful answers from internal models . . . 123
6.5.2 Experiment 2: ablation of the robot context selection . . . 124
6.6 Limitations and outlook . . . 126

**7 Conclusion 129**
7.1 Summary . . . 129
7.2 Derived works . . . 131
7.3 Discussion . . . 133
7.4 Open research directions and current trends . . . 134
7.4.1 Foundation model-powered knowledge acquisition and representation 135
7.4.2 Advanced interaction paradigms for shared control in assistive robotics 135
7.5 Closing remarks . . . 136

**Bibliography 137**

# 1

# Introduction

## 1.1 Motivation

The long-standing vision of assistive robots that aid humans is progressively being realized. Many research prototypes and a few commercial devices [164, 163, 126, 166] offer a hopeful outlook to millions of people around the world suffering from motor impairments[1]. For many of them with severe cases, simple activities of daily living [97] become impossible without assistive devices or caregivers.

Input devices for assistive robots, such as a joystick or an electromyography sensor array on the skin's surface [194], often have low-dimensional input spaces. Depending on their level of disability, users can achieve a relatively stable command of these interfaces, sometimes for up to three continuous degrees of freedom and an additional discrete grasping signal [196, 194]. In the worst cases, unfortunately, users will be unable to use these devices at all and will need to resort to discrete sensors, such as a head switch, a Sip-and-puff setup, or a Brain-Machine Interface [128].

But using a low-dimensional input device to control an assistive robot with many degrees of freedom can be challenging [78] and, in our own experience, frustrating. Consider the task of controlling a wheelchair-based mobile manipulator to open and go through a door, illustrated in Fig. 1.1 with the DLR EDAN system. Suppose users are using a 3D joystick, for example. In that case, they will need to switch between simultaneously controlling the robot end-effector (EE) translations, the three EE rotations, the two dimensions of the non-holonomic wheelchair, and the hand configuration for grasping [196]. This method is known as *mode switching*, and is not only cumbersome (because humans do not naturally represent movements in terms of end-effector positions in Cartesian space [33]), but is also prone to errors, and has been shown to require a high cognitive load [78]. It is thus undesired for

[1] Some statistics: in 2013, the prevalence of paralysis was nearly 5.4 million in the US [8], and the World Health Organization estimated that between 250 000 and 500 000 people per year suffered a spinal cord injury around the world [142].




**Figure 1.1:** The wheelchair-mounted mobile manipulator DLR-EDAN, traversing a door it just opened.

users.

A naive alternative would be for users to issue commands in natural language (e.g., "*open and go through the door!*"), and have the robot follow their orders. After all, decades of robotic research have led to significant developments in robot autonomy, including advancements in object perception, compliant manipulation, and task-and-motion planning. However, autonomy alone is not a suitable paradigm either: studies show that users with motor impairments express satisfaction with having control authority over the robot's movements, even if this leads to higher workloads [102, 25]. So how do we design user experiences for assistive robots?

#### A trade-off between *autonomous support* and *empowerment*: shared control

Aware of this problem, the research community actively develops frameworks where control during a task is shared between the user and the robot's autonomy, and thus coined the term "shared control". I use the following definition, which I expand in Chapter 2:

> **Shared control** is *"the control modality where robot control variables are jointly controlled by the human and the system [155], either proportionally, split along degrees of freedom of motion, or by means of constraints on the robot body"*. (Def.1.1)

Users are thus empowered to be in control of the movement, but are assisted by the robot

autonomy modules during the execution of a task. To open the door in Fig. 1.1, for instance, they can issue joystick commands to translate the EE, and the robot will assist with the EE orientation, the finger configuration, and the wheelchair driving [88][2].

A key feature of shared control is that users can execute tasks as they wish. This can mean, for example, that they can control the pace of the robot's movements and shape its actions to their own will. Consider Fig. 1.2, an example of a shared control situation where the task is to serve a drink. Shared control can help by pointing the end effector towards the bottle while grasping, or by keeping a bottle tip over a glass while pouring to avoid spilling. But it does not need to completely predetermine the approach direction or the amount of water that is poured. The usefulness of shared control, therefore, lies in that balance between autonomous support and user empowerment.




**Figure 1.2:** Examples of shared control tasks with the robot EDAN, where a user is serving water using a 3D joystick. The first three panels show the task of grasping a cylindrical thermos bottle, where the user is free to select the desired angle of grasping using constrained EE positions, which will be explained in detail in the later chapters of the thesis. In the fourth and final panel, I show the subsequent pouring task; here, the user can control the tilting angle of the bottle, deciding how much and for how long to pour water.

## 1.2 Cognitive shared control and its desired features

Aiming to improve shared control, this thesis considers artificial intelligence methods that enable a robot to process knowledge about its environment and the objects it contains. By *interpreting* this knowledge, the robot can provide users with support tailored to their specific situation. The robot can, for instance, know the *actions the environment objects afford*, whether such actions are *feasible in the specific context* (or not), and *the effects* these actions would have on said objects after being executed. Furthermore, as a consequence of understanding their actions, robots can *provide explainability and interpretability* to their users.

This need for contextual reasoning requires a robot to have *internal representations* of the world, its actions, the environment, and its goals. Robots with such internal representations of knowledge are referred to as "cognitive robots"[3] in the literature. Applying cognitive

[2] An example of a user performing this very task in shared control can be seen in the following video: https://youtu.be/pbL8kK8VXMo?feature=shared&t=117

[3] Formal definitions of cognitive robots are scarce. In psychology, cognition is defined as "*the brain's ability to acquire, process, store, and retrieve information*" by Khera and Rangasamy [101]. However, in robotics, "*the mean-*

robots to shared control results in what I call "cognitive shared control", where I propose the following definition:

> **Cognitive Shared Control** means *endowing shared control frameworks with the ability to represent, acquire, retrieve, and interpret knowledge, with the goal of providing flexibility, robustness, goal guidance, and explainability*. (Def.1.2)

As the definition above implies, cognitive shared control yields several desirable properties from the user perspective. In particular, the thesis explores how cognitive shared control addresses four important open research questions with respect to **four desired high-level features** of shared control frameworks illustrated in Fig. 1.3.

These four desired features — **flexibility**, **goal-guidance**, **robustness**, and **explainability** — correspond to one of the core contributions of this thesis, with cognition as their cornerstone. In the following subsections, I describe these features in detail: I (1) summarize the motivation for studying them; (2) derive research questions related to them; (3) summarize the specific solution to those research questions in the thesis (to be detailed in the remaining chapters), and (4) present a summary of the specific contribution:

### 1.2.1 Contribution 1: shared control should be flexible

**Motivation:** Users of assistive robots divide their time between their bed and their wheelchair[4], and may want to engage with the robot depending on factors such as workload, system trust, tiredness, or engagement. Therefore, shared control frameworks should be **flexible**, meaning that users can seamlessly allocate control to themselves or the robot on a given task. This is illustrated in Fig. 1.4, where the same task is conducted in shared and autonomy control modes.

**Specific research question:** How can we provide seamless transitions between shared control and robot autonomy?

**Proposed solution in the thesis:** Traditionally, switching between distinct levels of autonomy entails using different action representations, so the robot needs to re-plan on every

*ing of cognition has become broadened to include not only the one-way process of understanding a given situation, but also the process of creating motion with the functions of learning, memory, and motion controls as a signal loop*", as summarized by Shimoda et. al. [181].

[4]The preferences of our target group are very heterogeneous. Some users spend most of the day in the wheelchair, while others stay in bed for long parts of the day.

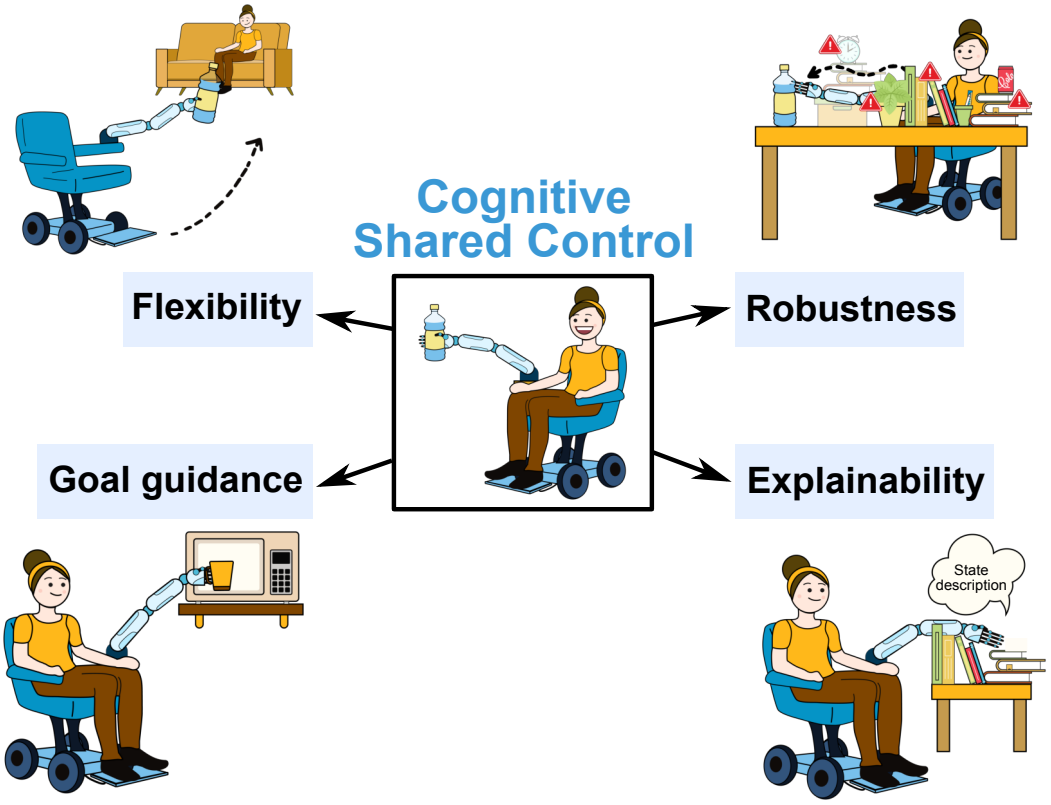



**Figure 1.3:** An illustration of four ***desired features*** addressed by cognitive shared control. Top left: the system should be **flexible**, such that users can decide how much autonomy to allocate to the robot, based on their temporal preferences. Top right: the system should be **robust**, avoiding obstacles in the environment and other kinematic limitations. Bottom left: the system should offer to **guide** towards **user goals**, offering support over long-horizon actions, for instance, to warm up a drink in a microwave. Bottom right: finally, the system should be **explainable**, meaning it should articulate explanations about its internal reasoning, to foster user trust. **Credits:** All illustration panels are an artist's depiction (by Adelina Flórez, © thesis author).

switch. Our approach addresses this technical challenge by utilizing a single action representation, introduced later in Section 1.3, and implementing the autonomy switch in the same space where the user provides input commands, such as the 3D joystick. More concretely, the module responsible for autonomously completing the task, known as the *automaton*, is provided with the same shared control interface as the user. As this control interface simplifies control, it is shown in **chapter 3** that a naive optimization-based algorithm is enough to solve a variety of daily living tasks.

**Summarized contribution:** *The Automaton*, a method for Shared control with integrated autonomy, i.e., for realizing seamless transitions between shared control and supervised autonomy, without the need to re-plan between transitions.




**Figure 1.4:** Shared control with integrated autonomy while opening and traversing a door. In our solution, the same action representation is used to support both control modes. That way, users are empowered to open the door in shared control, if they wish to do so, but they can also let the robot execute the task autonomously, even remotely, while operating the robot from another room.

### 1.2.2 Contribution 2: shared control should be robust

**Motivation:** Assistive robot failures may lead to user frustration. For instance, poor device performance is a known factor related to assistive technology abandonment, with reliability being an important aspect for users [146]. As in robotics in general, many aspects of the daily operation of a shared control robot can produce failures.

A particularly relevant pitfall is driving the robot into *kinematic limitations*, a term used to describe a variety of movement limitations that arise during teleoperation of kinematic chains due to obstacles, virtual walls, joint limits, singularities, or reduced manipulability [39]. In this context, shared control frameworks should be **robust**, meaning that while they support the user in a task, they should identify (and preclude) motions that could lead to kinematic limitations. This is illustrated in Fig. 1.5, where a grasping mug task fails due to proximity to several kinematic limitations.

In order to prevent the user from falling into kinematic limitations, previous works have proposed methods that apply constraints locally *during* the movement (e.g. [5, 177, 176, 169]). However, a task may still be unfeasible if conflicting task constraints cannot be resolved, for instance, when simultaneously applying task-related and kinematic-limitation-related constraints (as shown in Fig. 1.5, for avoiding a safety virtual wall and a joint limit). Therefore, it would be beneficial if the feasibility of the task could be checked *before* the user starts executing it.

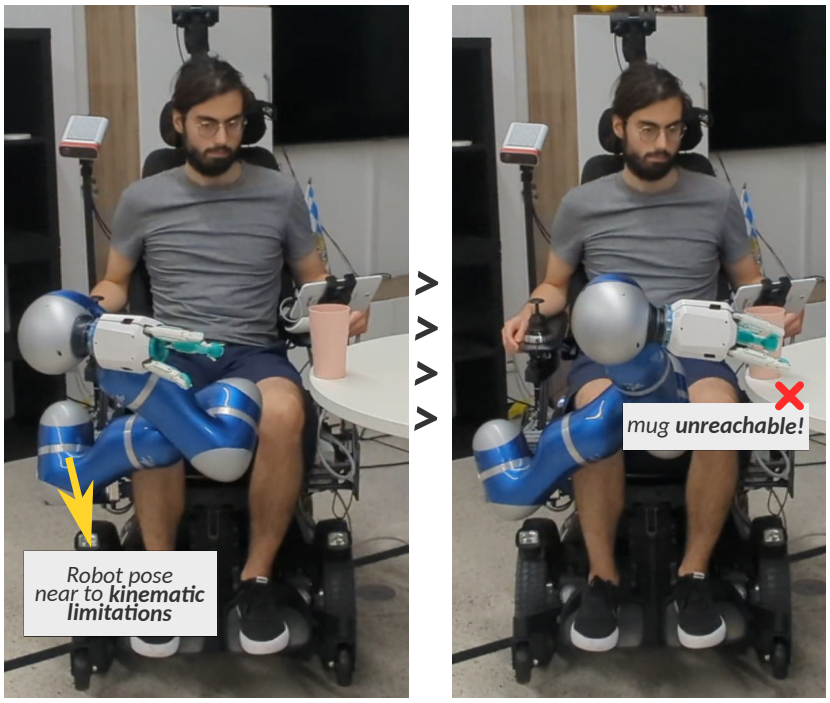



**Figure 1.5:** A user fails to grasp a mug in shared control. The robot's initial configuration (left image) is near kinematic limitations due to a joint limit and a virtual wall that protects the user. Such a situation may not be immediately apparent to the user, who may naively start a task (right image). However, as the task-related constraints are not enforced, the task fails without an immediately apparent reason. Credits: Supplementary media from [38]. © 2024 IEEE.

**Specific research question:** Rather than running into kinematic limitations during a shared control task, can kinematic limitations be predicted (and therefore precluded) before executing the task? And once a task is deemed feasible and started, how can the freedom-of-movement principle in Fig. 1.2 be ensured while preventing kinematic limitations during the task?

**Proposed solution in this thesis:** Our method solves this problem by offering two solutions:

1. by conducting feasibility checks to detect kinematic limitations *before* shared control for a task is activated, using a kinematic twin of the robot and models of the environment.

2. by handling kinematic limitations, either by precluding execution of an infeasible task — i.e. in case no path was found which achieves the task — or by finding a *task region* (defined in Section 1.3) where the task is deemed feasible, and *refining* the task constraints to keep the robot in this region. Thus, users are granted freedom of movement while respecting kinematic limitations and safety considerations, and in some cases, this would prevent them from discovering in the middle of the task that it is not feasible at all.




**Figure 1.6:** A long-horizon task with goal guidance in shared control. The user's explicit goal is to place the mug inside the microwave, perhaps to warm up tea. Goal guidance, in this context, refers to assisting at every step towards the goal, including sequencing the subtasks for opening the microwave, grasping the mug, and placing the mug. However, it also means enabling freedom of movement within those subtasks, as illustrated in Fig. 1.2: here, the user can select, for instance, the placement of the mug within the microwave.

**Summarized contribution:** A method for performing feasibility checks on a digital twin of the robot before a shared control task, which further refines task-specific shared control constraints.

### 1.2.3 Contribution 3: shared control should provide guidance towards a goal

**Motivation:** In a human-computer interaction context, so-called *goal-guided interaction* can support users who do not want (or do not have the ability or time) to handle a learning curve, particularly occasional and novice users [41]. Therefore, shared control frameworks should reason in complex environments to **guide** users towards their **goals**. Furthermore, goal guidance can also provide more flexibility, as users are given the choice between expressing minimal goals or providing high oversight for the specific actions the robot takes [148]. This flexibility could therefore lead to an improved user experience, as has been shown in other fields [40].

The technical challenge for such a system is to enable freedom of movement within an action, as illustrated in Fig. 1.2, while guiding the user to a specific goal. Such a system is illustrated in Fig. 1.6.

**Specific research question:** If a sequence of shared control sub-tasks is required to achieve a goal, how can this sequence be planned autonomously and then executed without limiting

freedom of movement?

**Proposed solution in this thesis:** To enable goal-guided reasoning, we introduce a task planning method that generates a sequence of sub-tasks for the user to follow in order to reach a user goal. This is illustrated in the microwave example in Fig. 1.6, where the user has instructed the goal of placing a mug lying on the table inside the currently-closed microwave. The system guides the user to first open the microwave, then grasp the mug, and finally place the mug inside the microwave.

Central to this contribution is a novel object-centric action representation, called Constraint-Action Templates (CATs), which is explained in more detail in Chapter 5. CATs leverage a combination of *declarative knowledge*, useful for symbolic planning, with *procedural knowledge*, which contains the necessary information for the robot to provide the user with freedom of motion.

**Summarized contribution:** Constraint-Action Templates, a novel hybrid action representation that enables task planning for shared control, and a method for task planning of shared control tasks.

### 1.2.4 Contribution 4: shared control should be explainable

**Motivation:** For users of any artificial intelligence (AI) technology, explainability and transparency are central to fostering trust [55, 197]. Therefore, shared control frameworks should be **explainable**, meaning the robot should be able to convey its internal knowledge, beliefs, and rationales to users on demand.

Recently, the advent of foundation models (FMs) such as Large- and Vision-Language Models (LLMs/VLMs) has enabled machines to parse and output realistically-sounding natural language, which has enormous potential to increase the interaction capabilities of assistive robots. In the explainability context, the technical challenge consists of grounding the output of FMs in the internal knowledge of the robot, rather than simply providing an answer based on knowledge derived from their weights, i.e, general but unspecific knowledge from a large set of (most likely internet-based) training data. This is illustrated in Fig. 1.7, where a user asks his assistive robot a question about the present status of its internal beliefs.

**Specific research question:** Can the robot utilize foundation models to offer a natural language interface that, when necessary, explains the internal beliefs of the robot and the shared control system?

**Proposed solution in this thesis:** We introduced *RACCOON* (**R**obots **A**nswering questions **C**ognizant of **CO**ntextual c**O**g**N**itive modules), a framework for question-answering robots

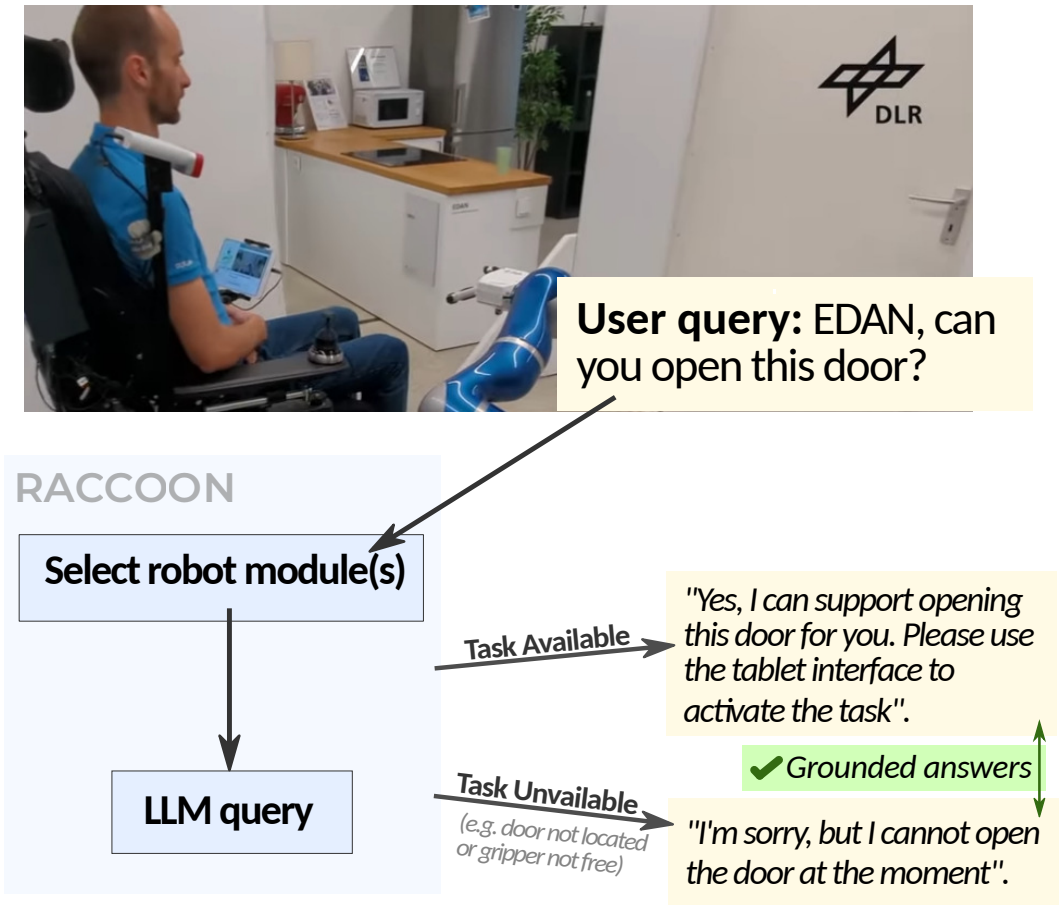



**Figure 1.7:** A summary of our **explainability** method, RACCOON. If the user asks a question about EDAN's current capabilities, the answer lies within the system's beliefs, so RACCOON must retrieve the information from there. In this case, whether EDAN can open or not the door depends on variables such as the door handle being located or the state of the gripper (e.g., whether it is free or not). **Credits:** the bottom part of the image is a simplified illustration from [35], with the longer version shown in Chapter 6 (© 2025 IEEE). The EDAN user illustration at the top is licensed under a Creative Commons Attribution license (reuse allowed), as cited in [156].

with pre-existing knowledge modules. RACCOON selects the modules that can answer user questions, and then extracts information from these modules using summaries of the state of these modules, i.e., descriptions of the robot state in the form of natural language. To answer user questions, RACCOON uses these summaries as a prompt for a Large Language Model, enabling flexible question-answering while keeping these answers grounded in the robot's beliefs. An overview with example outputs is shown in Fig. 1.7,

**Summarized contribution:** *RACCOON*, a method for query-answering for robots with pre-existing modules, leveraging an LLM to create explanations grounded in the internal beliefs of the robot.

### 1.2.5 Summarizing example

To support the user with opening a drawer, for example, a cognitive robot like EDAN needs to represent the concept of a drawer handle; it needs to know that drawers can afford to be opened if they are closed; it needs to locate the drawer and assert if it is opened or closed; and it needs to represent the necessary procedures to open the drawer, step by step.

Critically, these cognitive processes can further support shared control, making it more useful to the target users. Interpreting and using the acquired knowledge about the drawer, for instance, the robot can provide a shared control framework that can (*i*) plan a multi-step task involving the drawer, like grasping an object known to be inside of it (thus providing **goal guidance**); (ii) simulate the action to check the feasibility of opening the drawer in the current environment (providing **robustness**); *(iii)* give explanations to users, for instance, if they want to learn why the drawer cannot be opened (providing **explainability**); and *(iv)* execute a drawer opening task in both shared control and autonomy mode, and allowing the user to seamlessly transition between both (providing **flexibility**).

## 1.3 Knowledge representation for cognitive shared control

To act in the real world, robots need to *model* both the world and the actions. Thus, two types of models are central to the cognitive shared control framework of the thesis: a model of **actions** ("*action model*") and a model of the **world** ("*world model*"). I illustrate them in Fig. 1.8 and describe both of them below.

**Action model:** Leidner provides the following definition:

> "***Actions*** *relate motions to effects. An explicit* ***representation*** *of this relation enables a robot to reason about motions in a goal-oriented manner.*" Leidner [117]. (Def.1.3)

Thus, modeling actions requires two elements, illustrated in Fig. 1.8 (left):

- a *procedural* component, i.e., a geometrical description of the motion. In shared control, this description cannot be limited to just one robot path; instead, it must encode different ways in which the object affords this action to occur, empowering users to shape the actions according to their preferences (as seen in the initial grasping and pouring examples in Fig. 1.2).
- a *declarative* component, i.e., providing a high-level factual summary of the preconditions and the effects of the motions in the involved objects.

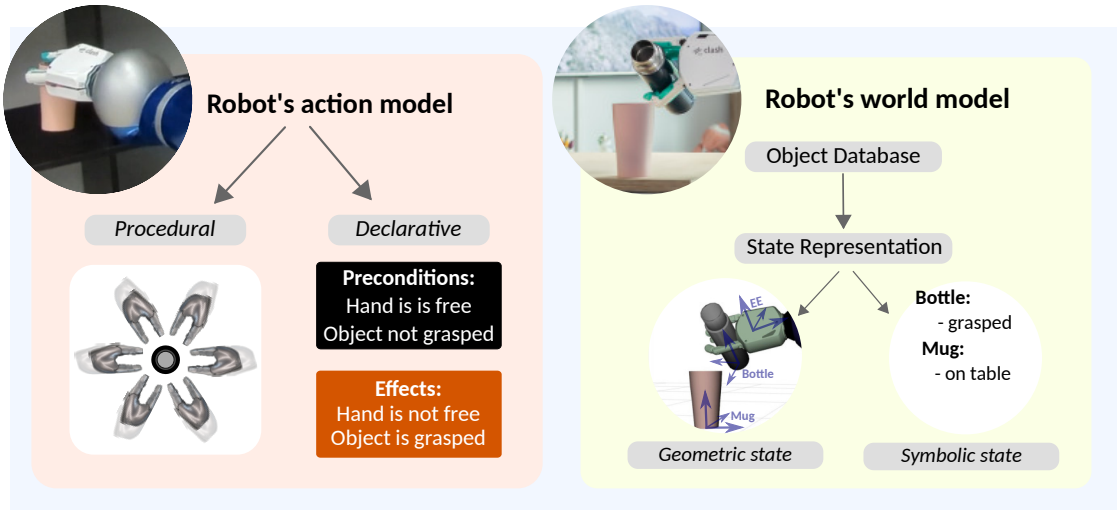



**Figure 1.8:** Illustration of two different models used with our framework. The depictions of either model are only illustrative and not complete. **Left:** representation of the robot's action for grasping a mug. It describes a procedural part, including a description of the mug's vertical symmetry for grasping, and a declarative part, containing a symbolic description of the preconditions and effects of a grasping action. **Right:** illustration of the robot's world model while pouring. It includes a geometric state, which contains the transform tree of objects (displayed as a set of blue coordinate axes, stored as matrices in the world model), and a symbolic part, containing the symbols of the current state. **Credits:** the left panel contains design elements from [37, 38] and the right panel was modified from [69].

My work is based on the Shared Control Templates (SCTs), a procedural representation form previous work [155, 154]. SCT actions are a sequence of state machine states, where each state defines user command mappings and End-Effector constraints described in detail in Section 2.3. Over the course of the thesis, we extended the SCTs with two complementary representations:

- Refined Task Space Regions (rTSR), a procedural representation inspired by the Task Space Regions from the motion planning literature [21]. We adapt the rTSRs in a shared control context to perform feasibility checks over the SCTs, explained in detail in Chapter 4.
- Constraint-Action Templates (CATs), a representation that adds declarative knowledge to SCTs to enable goal-oriented reasoning, and described in Chapter 5.

**World model:** Our world model is the so-called *world state representation* (WSR), adapted from previous work [117]. This state representation contains two types of information:

- Geometric knowledge in the form of a transform tree. The robot may know the position of the mug with respect to the origin. It may also be known that the bottle is grasped, such that the position of the bottle with respect to the robot hand is fixed.
- Symbolic knowledge as a list of symbolic statements (called *predicates*) with descriptions of the state of single objects (e.g., "the bottle is grasped", "the microwave is open")

or of relations between them (e.g., “the mug is on the table”).

There is an important property of these predicates: even if they are modeled in formal languages (such as the Planning Domain Definition Language, PDDL [64]), the symbols are usually very close to natural language (for example, the predicate “`open ?lab_microwave`” is very similar to the English statement “the lab microwave is open”), and thus are suitable for explainability frameworks based on LLMs, such as the one presented in Chapter 6.

Robots that interact with the environment also need to leverage pre-existing object knowledge, like the length of a specific drawer handle or the height of a bottle. For this purpose, we use an Object Database (ODB) that informs the WSR, following the paradigm outlined in [117]. Finally, the state representations can be informed by the robot’s perception pipeline, both in the symbolic [109] and the geometric [188] cases.

## 1.4 Related work

While based on the related work sections of the main thesis papers[5] ([36, 37, 38, 35]), the goal of this section is to provide a holistic review of the state of the art. It contains a recent revision of the literature with novel content and papers up to 2025.

**Historical note:** The very first demonstration of a human teleoperating a robot dates back to the 1940s and 50’s, when Raymond C. Goertz researched robotic prototypes for humans to manipulate radioactive material behind a safety wall [138]. This technology evolved in the following decades, with a significant milestone in the ROTEX space experiment in April 1993. There, humans on ground and in space could operate a robot manipulator on the Spacelab (a laboratory inside the Space Shuttle Columbia) [80, 138]. ROTEX already showcased the principles of cognitive shared control, i.e., “*sharing the control variables between operator and machine, based on knowledge processing*”: As the round-trip delay between the SPACELAB and the ground was 6–7 seconds, the robot in space used local autonomy based on task knowledge to aid the teleoperator, thereby enabling the completion of service tasks in space [162, 138]. The same principle has been applied in various domains, ranging from remotely operated vehicles in the depths of the sea [111] to surgical robots [72].

**Knowledge representations for shared control:** To give a human operator support in a shared control task a robot must not only represent *how to perform an action with a set of objects*, but also *how this action can vary by the execution of the user*. In more concrete terms,

[5] Some paragraphs have been taken literally and embedded in the new text, whenever corresponding to literal text blocks: © 2021-2025 IEEE

shared control must represent the *affordances*[6] offered by the objects. A thermos bottle such as the one in Fig. 1.2, for instance, affords to be grasped anywhere across its circumference and height, and it should ultimately be the user who decides where. While object-centric affordance representations for autonomous robots have been investigated for a while (e.g., for cylindrical objects [46, 192]), it is only recently that they gained attention in the assistive robotics community, yielding frameworks like the Shared Control Templates [155] or Learned Latent Actions [124].

A related representational issue is how to enrich these geometric affordances with declarative knowledge about the task, namely that a robot does not only represent *how to pour a drink* but *what it means to pour a drink*. The robot then needs to know what objects are involved (e.g. "*pouring a drink requires two containers*"), what state needs to happen before the action takes place ("*the robot can only pour if it has grasped one container first*"), and what effects the action has on the environment ("*the liquid will be transferred from the grasped container to the other one*"). These representations may be implicitly learned, for instance, by using language-informed latent actions [95] or explicitly defined, by wrapping skill primitives with formal languages for planning [117].

**Adaptable autonomy with goal guidance:** A key area of research in the human-robot interaction literature focuses on deciding the allocation of autonomy between the user and the robot. After all, the level of autonomy of a robot needs to be tailored to the specific requirements of users [89].

As early as 1989, Hayati and Venkataraman designed a robotic system with so-called Traded Control[7] capabilities [75]. Ever since, many works have explored adaptive autonomy allocation in a teleoperation context, for instance, in industrial [11] and space [119] domains. However, many of these rely on haptic feedback [4], which is not very common in assistive robotics. Instead, a large number of assistive robotics works were inspired by a landmark paper by Dragan and Srinivasa [2], which consists of generating a robot policy that solves a specific task using a motion planner and *blending it* with user commands mapped to the end-effector space. This formalism has been widely used in diverse works at various levels of autonomy [93, 66, 90]. However, the blending formalism in [2] requires the robot to map the user commands to the same space as the robot policy (often expressed as EE poses), thus creating the need to combine two different representations. More critically, if the user is in complete control and trades control to the robot, it may force it to re-plan its policy, because the motion plan may be outdated (or because the robot cannot know how the environment changes while the human is in control [105]). As users may like to freely adjust the level of

[6] *"Affordances are possibilities for action provided to an animal by the environment — by the substances, surfaces, objects, and other living creatures that surround it"* Rietveld and Kiverstein [161] paraphrasing Gibson [65].

[7] The problem of switching between Teleoperation, Shared Control, and autonomy is often referred to as *Traded Control*. However, other terms such as *shared autonomy*, *adaptable autonomy*, and *scalable autonomy* [119] are also used.

autonomy on the fly depending on factors such as workload, system trust, or engagement at the moment, this dual representation may not always provide seamless transitions.

This problem becomes even more complex when considering a sequence of tasks rather than a single one, as illustrated in the liquid serving example in Fig. 1.2. Research about creating a sequence of shared control tasks towards a goal is rare. However, task sequencing is a well-known problem in robot autonomy [117], and there is a need for goal-guided interaction in Human-Computer interaction, particularly for non-expert users [41]. While planned shared control systems exist [207, 63, 17], they usually do not plan an explicitly long sequence of actions while providing the option to switch to supervised autonomy.

**Avoiding kinematic limitations:** Section 1.2.2 already introduced the concept of *kinematic limitations*, i.e., a variety of movement-related issues that arise during robot operation due to obstacles, virtual walls, joint limits, singularities, or reduced manipulability [39]. A distinct line of work in the shared control literature consists of aiding operators to prevent, avoid, or recover from these limitations.

A common and effective method for signaling proximity to kinematic limitations during shared control is the use of haptic cues from force feedback [5, 177, 176]. However, while this modality may suffice for fields such as remote surgery or telemaintenance, it is unlikely to be directly applicable in assistive robotics, as the specific hardware for haptics and the required high-dimensional (6D) control are challenging for many target users with motor impairments to use. An alternative approach is to provide the user with constrained motions, either with a static environment (e.g., created using perception [180, 155]) or with reactive approaches during task execution [5, 177, 176, 193, 169]. While at least one of these approaches has been used in a research prototype for an assistive system (utilizing a low-dimensional interface [169]), they have the disadvantage that users are unaware of the feasibility of a task before executing it. In other words, they must ascertain the task's feasibility only *during* its execution.

In principle, motion planning could compute one feasible path that solves the task, and guide the user along this predetermined path [29]. However, this would not satisfy the freedom-of-movement requirement for user empowerment, like in Fig. 1.2.

**Task- and Motion Planning in Robotics:** Research in simultaneously generating high-level symbolic plans and low-level geometric plans is referred to as hybrid planning or, more recently, task and motion planning (TAMP). TAMP paradigms extend many robotics applications, such as in space [127, 165], industrial manufacturing [86], search-and-rescue [14], BMIs to control intelligent robots [107] and wheelchairs [122], service [117] and household robotics [61]. Monitoring the execution of TAMP plans can be done with reasoning and backtracking algorithms, such as the Cognitive Robotics Abstract Machine [16].

The symbolic parts of the plan commonly involve first-order languages, such as PDDL.

They enable the robot to reason about its actions on a declarative level: what is the effect of my action? What is the state the robot needs to be in in order for this action to be possible? Picking up an object from the fridge, for example, may require the PDDL predicate `opened fridge`. Closing the fridge will cause the PDDL predicate `not(opened fridge)`.

The geometric part of the plan involves determining the robot motions that can move the robot arm from a position $\mathbf{T}^0_{\text{EE-A}}$ to a position $\mathbf{T}^0_{\text{EE-B}}$, while avoiding obstacles and other kinematic limitations. More interesting for us are the *representations* from where these frames of interest are obtained. These arise from representing objects and their affordances. One example is the Task Space Regions (TSR) [20, 21], a representation of the possible ways in which a task can occur, i.e., a region in SE(3) that defines the task frames. For example, as a cylindrical container is symmetric around the axis, its grasp frame can be located anywhere around it. The TSR representation is one of the building blocks of my work, and is described in depth in Section 4.1.1.

Constraint representations based on geometric features, like the TSRs, have been used extensively to generate robot behaviour, for instance, a symbolic representation of a pancake-flipping robot [12] and a task frame formalism [31] for modeling robotic contacts in terms of frames of interest within a task. Other examples include allowable task manifolds based on demonstrations [193], shape-templates [79], grasp manifolds [71], a system of task-related constraints supervised by a human [145], affordance templates [74], and other object-centric approaches that exploit the object symmetries [46, 192].

In recent times, methods that employ Large- and Vision-Language Models (LLMs/VLMs) have been developed to enhance robot representations and planning algorithms. For instance, LLMs can generate symbolic representations of task affordances [26] and task-related constraints [84]. Likewise, VLMs can inform the world model with information about the robot's state and surroundings [83, 45, 98]. In the assistive robotics community, for instance, a method utilizes VLMs to inform the robot of the contents of a plate, and an LLM to perform task planning directly, taking into account user instructions and preferences [91].

**Explainability in robotics:** In the context of robotics, I adopt the following definition:

> **Explainability** refers to adding *"system modules that explain their internal workings (such as their intents, policies, and plans) to humans as their target."* Wallkoeter et. al. [197] (Def.1.4)

This definition highlights the distinction between traditional approaches to explainability in deep learning and explainability in robotics. While data-driven methods contain implicit, often inaccessible representations, higher-level decision modules require explicit knowledge representations (see e.g. Section 2.2). So explainability in robotics is not only about answering *why did the object detector fail to detect a bottle*, but, more generally, to explain to the user

that *the picking action of the bottle failed, i.e., the bottle cannot be picked up because the robot has failed to detect it*.

To clarify this distinction further, in our previous work by Stulp et. al. [186], we introduced a philosophical analogy for this problem:

- Explainability in data-driven deep learning may be considered "Freudian" (after the work of Sigmund Freud), as the knowledge responsible for decision making is implicit inside the model, and must be extracted post-hoc by specialist algorithms that work directly with the weights of the model [186].

- Explainability in cognitive robotics may be considered "Wittgensteinian" (after the work of Ludwig Wittgenstein), as the knowledge responsible for decision making is usually stored explicitly inside the robot models [186] in *interpretable* symbols, although algorithms must still extract it. Therefore, a prerequisite for Wittgensteinian-like explainability is that the symbols stored inside the robot share the meaning of the objects they represent, and that human users also understand this meaning. For instance, if a robot wants to explain that it cannot locate a bottle, the explanation will only succeed if its internal representation for the bottle is a symbol understood by the human, such as `bottle` or `ikea_bottle13`. If the representation of the bottle is a symbol that is unknown to the human or the explanation module (e.g., a Large Language Model), such as `blob2443`, the explanation process will likely fail [186][8].

**Explainability requirements from the social sciences:** Explainability helps foster trust in AI systems [55]. Without this trust, users may abandon these applications. As Miller notes:

> *"The running hypothesis is that by building more transparent, interpretable, or explainable systems, users will be better equipped to understand and therefore trust the intelligent agents"* Miller [131] citing [130, 76, 44]

In a landmark paper [131], Miller explains the characteristics of explanations in AI from a social science perspective. One of the most important remarks is that explanations are *contrastive*, i.e., that explanations are sought to respond to counterfactual situations of the form "`why has event P happened *instead of* event Q?`". For that reason, the AI planning community in robotics has been focusing on generating answers to this template of queries: "`why did you decide on P / why did P happen and not Q?`"(e.g. [43, 27, 106, 183, 56]).

Other methods for explainability in robotics focus on accumulating and modeling knowledge during decision processes [173], or investigate appropriate interfaces for robot users,

[8] A not-so-distant example is that the symbol representing our mug in Chapter 5 is `red_mug`, but (as a reviewer once noted) the actual object appears pink. This highlights a representational challenge for explainability.

particularly as some modalities (such as visual feedback) are inherently more explainable to users [210].

Finally, beyond contrastive questions, Miller offers two further insights about AI explanations:

- Explanations are selected in a biased manner; the explaining agent must choose appropriate levels of abstraction for their explanation, as humans do under cognitive biases [131].

- Explanations are inherently social, presented by the explainer relative to beliefs of the explainee [131].

**Interfaces for explainability:** A final related research line concerns how the robot's internal state is communicated to the user. Zolotas and Demiris argued that augmented reality (AR) and head-mounted displays are appropriate interfaces for users of assistive robots. They proposed a paradigm for explainable shared control based on visual feedback [210].

Natural language via speech is a promising interface for robotic explainability systems. However, one potential issue with natural language explanation methods is that they require matching a pre-existing template, rather than parsing free natural languages [184]. According to Sado et. al.:

> *"Adding a rich [natural language processing] system to existing frameworks would significantly benefit [explainable Goal-driven artificial intelligence] in its usefulness and applicability" Sado et. al. [170].*

A particularly unexplored avenue of research (at least until the submission of [35]) concerns natural language interfaces for cognitive shared control systems. One alternative is using foundation models (FMs) such as Vision- and Large Language Models (VLMs/LLMs), which have powered a recent wave of robotic demos where robots provide answers to arbitrary human questions about past [51, 58] and present [160, 175, 103, 58] situations concerning the robot. However, naively prompting a VLM to process the camera images directly and take on the persona of the robot may give the impression that the robot is speaking, but would not take into account the robot's internal state during the interaction. Thus, VLMs alone do not make the robot's cognitive systems more explainable. There is thus a need for methods that ground the FM answers in the knowledge stored inside the robot. One promising avenue to achieve this is Retrieval-Augmented Generation (RAG). This well-established technique supports FMs answering knowledge-related tasks by *augmenting* (via prompt engineering) their answers with query-related context from the knowledge base [118, 96].

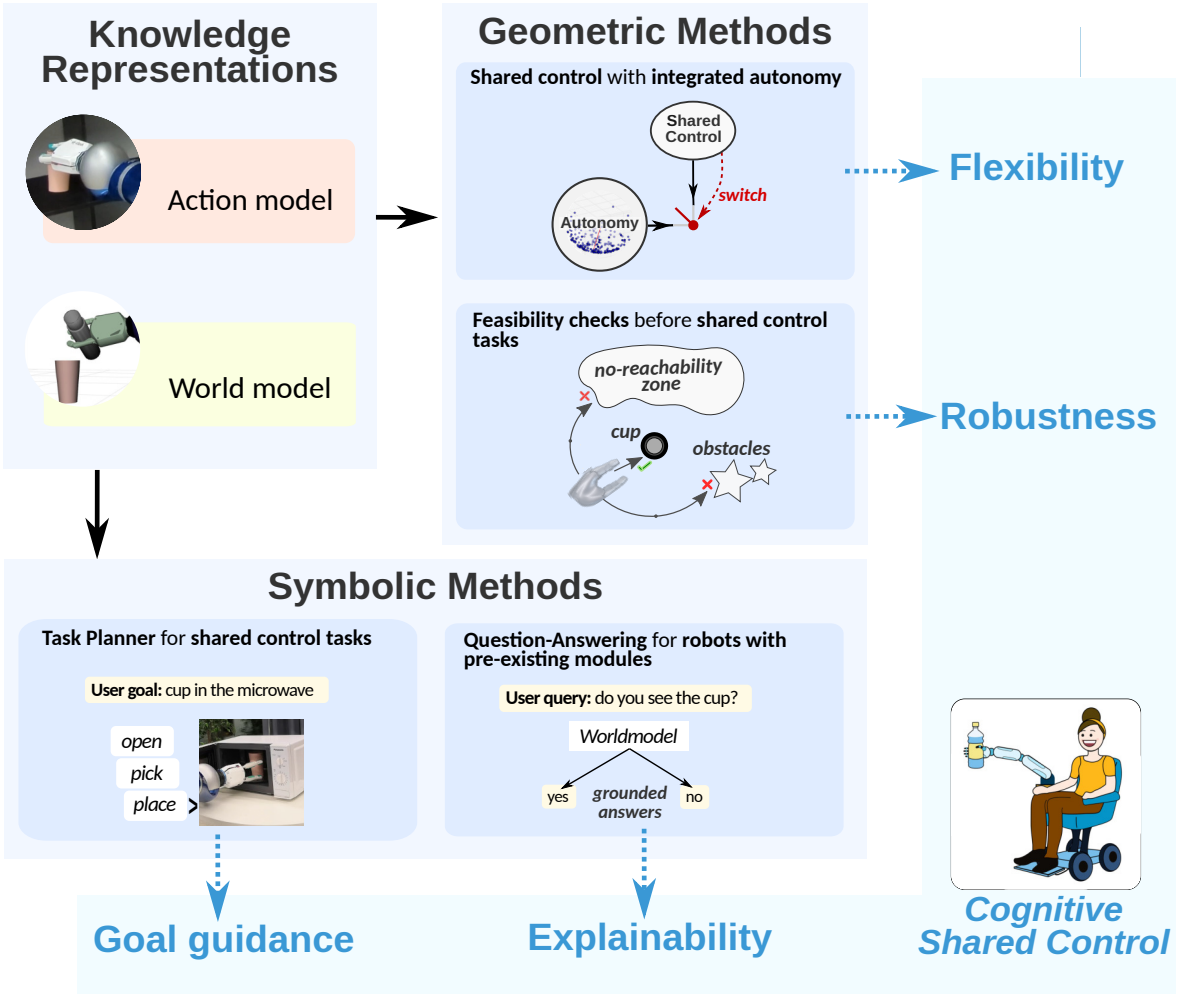



**Figure 1.9:** A *cognitive shared control* framework to empower users of assistive robots. The foundation is the robot's knowledge representation, encompassing models of its environment and the actions it can perform. These enable the robot to reason and plan on both symbolic and geometric levels, and the thesis introduces four core methods, two on each level. Each of these methods relates to one of the desired features of cognitive shared control, displayed in the outer panels. **Credits:** the sources of some of the boxes (world model, geometric models, and symbolic models) were taken and modified from several papers [69, 36, 37, 35, 38]. The illustration in the lower right is an artist's depiction by Adelina Flórez (© thesis author).

## 1.5 Summary: statement of contributions

My thesis contributes to the state of the art with a **cognitive shared control framework** shown in Fig. 1.9. Said framework is based on pre-existing robotic world and action models from related work [155, 154, 117]. It provides:

1. The *automaton*, a method for shared control with integrated autonomy, i.e., for realizing seamless transitions between shared control and supervised autonomy, without the need to re-plan between transitions. The automaton provides **flexibility**.

2. A method for performing feasibility checks before shared control tasks. It enables the user to ascertain if a task will fail before starting it, and otherwise refines the shared

control constraints to prevent failures during the task. This method provides **robustness**.

3. A method for task planning shared control tasks, enabling goal-guided interaction with the assistive system. This method provides **goal guidance**.

4. *RACCOON*, A method for query-answering for robots with pre-existing modules, leveraging an LLM to create explanations grounded in the internal beliefs of the robot, and thus making such robots more explainable. RACCOON provides **explainability**.

Furthermore, this thesis:

- Coins the term *cognitive shared control* as defined in Def.1.2.
- Provides two complementary action representations for shared control, the *refined Task-Space Regions* and the *Constraint-Action Templates*.
- Provides a survey of shared control in assistive robotics, including a taxonomy of shared control works (in Chapter 2).
- Reports experimental evaluations with the assistive wheelchair-mounted mobile manipulator EDAN and the assistant humanoid robot Rollin' Justin, including demonstrations in various daily living situations and tasks. These examples include grasping, pouring, opening drawers, doors, and microwaves, among others.

## 1.6 Thesis outline

The thesis structure and the relevant papers are shown in Fig. 1.10. As shown in the figure, the thesis is divided into seven chapters.

**Chapter 2** presents the thesis's fundamental background knowledge, with an emphasis on representations from previous work or from publications in which I am not the primary author (e.g., the Shared Control Templates and the World State Representation). Background information that is related to all chapters (e.g., Shared Control Templates) has been included in this chapter; in contrast, relevant background information only to one specific chapter is provided at the beginning of said chapter.

**Chapter 3** contains the algorithm and experimental evaluation of shared control with integrated autonomy. **Chapter 4** describes the first of the new action representations, the Refined Task-Space Regions, and further explains how they are used for checking the feasibility of shared control tasks. **Chapter 5** introduces the other new representation, the Constraint-Action Templates, and shows how they are used for task planning of shared control tasks. **Chapter 6** presents the query-answering method for robots with pre-existing modules. Finally, **Chapter 7** provides a summary, existing extensions, and a conclusion.

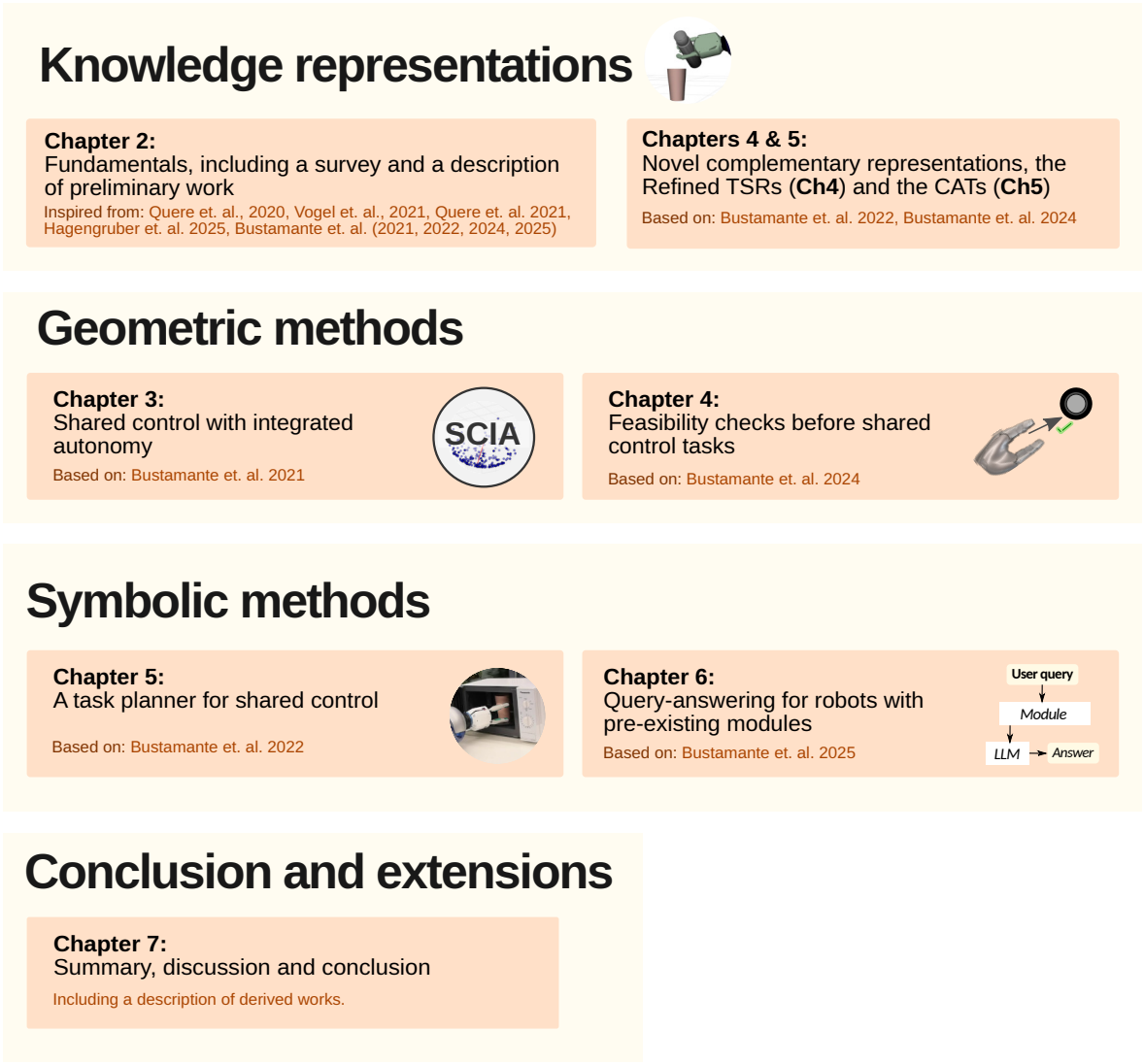



**Figure 1.10:** Thesis structure, and papers used as base of the chapters.

### 1.6.1 Publications note

#### Main set of publications

The thesis is supported by a set of publications, of which I am the first author on four. Some portions of those first-author publications (as shown in Fig. 1.10) are used as the base for the thesis chapters. These publications are:

- *Bustamante et. al. 2021:* **S. Bustamante**, G. Quere, K. Hagmann, X. Wu, P. Schmaus, J. Vogel, F. Stulp, and D. Leidner, "Toward Seamless Transitions Between Shared Control and Supervised Autonomy in Robotic Assistance," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3833-3840, April 2021. (Base for **Chapter 3**).

- *Bustamante et. al. 2022:* **S. Bustamante**, G. Quere, D. Leidner, J. Vogel, F. Stulp, "CATs: Task Planning for Shared Control of Assistive Robots with Variable Autonomy,"

2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022, pp. 3775-3782. (Base for **Chapter 5**).

- *Bustamante et. al. 2024:* **S. Bustamante**, I. Rodriguez, G. Quere, P. Lehner, M. Iskandar, D. Leidner, A. Dömel, A. Albu-Schäffer, J. Vogel, F. Stulp, "Feasibility Checking and Constraint Refinement for Shared Control in Assistive Robotics," in IEEE Robotics and Automation Letters, vol. 9, no. 9, pp. 8019-8026, July 2024. (Base for **Chapter 4**).

- *Bustamante et. al. 2025:* **S. Bustamante**, M. Knauer, J. Thun, S. Schneyer, A.Albu-Schäffer, B. Weber, F. Stulp, "RACCOON: Grounding Embodied Question-Answering with State Summaries from Existing Robot Modules," 2025 International Conference on Robotics and Automation (ICRA) in Atlanta, GA, USA, 2025, (Base for **Chapter 6**).

**Note on the usage of the papers as chapter base:** Although each of these main papers constitutes the base of one chapter, the text has been revised significantly. Not only have the chapters been improved and explained better, but also new content (such as figures, in-depth text, novel concepts, or anecdotal evidence of the methods' usage beyond the original papers) has been added.

Please refer to the abstract of each chapter for more information, as well as for the respective copyright notices.

#### Relevant publications as second or later author

Relevant publications as a second or later author during my PhD are:

- *Quere et. al. 2020:* G. Quere, A. Hagengruber, M. Iskandar, **S. Bustamante**, D. Leidner, F. Stulp. J Vogel, "Shared Control Templates for Assistive Robotics," 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 1956-1962, doi: 10.1109/ICRA40945.2020.9197041.

- *Vogel et. al. 2020:* J. Vogel, A. Hagengruber, M. Iskandar, G. Quere, U. Leipscher, **S. Bustamante**, A. Dietrich, H. Höppner, D. Leidner, A. Albu-Schäffer, "EDAN: An EMG-controlled Daily Assistant to Help People With Physical Disabilities," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 4183-4190.

- *Quere et. al. 2021:* G. Quere, **S. Bustamante**, A. Hagengruber, J. Vogel, F. Steinmetz and F. Stulp, "Learning and Interactive Design of Shared Control Templates," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 2021, pp. 1887-1894, doi: 10.1109/IROS51168.2021.9636047.

- *Hagengruber et. al. 2025:* A. Hagengruber, G. Quere, M. Iskandar, **S. Bustamante**, J. Feng, D. Leidner, A. Albu-Schäffer, F. Stulp, J. Vogel, "An assistive robot that enables

people with amyotrophia to perform sequences of everyday activities," Accepted to Scientific Reports.

During the final part of my PhD, I have contributed to two papers in a supervisory role (i.e., as the last author or as an author with equal contribution to the last author). These papers are briefly mentioned in the Conclusion, in **Chapter 7**:

- *Knestel et. al. 2025:* P.Knestel, L.Mayershofer, A.Köpken, N.Batti, F.Lay, T.Bachmann, S.Brinkman, J.Butterfass, E.DenExter, W.Friedl, T.Gumpert, X.Luo, A.Manaparampil, T.Mai, A.Raffin, F.Schmidt, A.Schmidt, D.Seidel, L.Schurmann, N.Wenzinger, B.Weber, A.Albu-Schäffer, A.Bauer, D.Leidner, R.Luz, P.Schmaus, F.Stulp, T.Krüger, **S.Bustamante***, N.Lii*., "NealAI: a RAG-based chat assistant supporting command of a robotic team from the ISS," accepted for presentation at the 2025 International Conference on Space Robotics (iSpaRo). *N. Lii and S. Bustamante share equal contributions in a supervisory role.

- *Besch et. al. 2025:* B.Besch, J. Thun, T. Mai, M. Huff, J. Vogel, F. Stulp, **S. Bustamante**, "A Model Reconciliation Approach for Collaborative Recovery Behavior in Explainable Shared Control," Under review for a conference at the time of the thesis submission.

#### Other papers or relevant contributions

During the thesis, I contributed to the following workshop papers:

- F. Stulp, A. S. Bauer, **S. Bustamante**, F. S. Lay, P. Schmaus, D. Leidner "Explainability and knowledge representation in robotics: The green button challenge," presented by F. Stulp at the Explainable Logic-Based Knowledge Representation (XLoKR 2020) workshop at KR 2020 (17th International Conference on Principles of Knowledge Representation and Reasoning, September 12-18, 2020, (virtual)).

- **S. Bustamante** et. al., "Feasibility Checks for Safe Shared Control with Variable Autonomy in Assistive Robotics," presented by me at the workshop in Intelligent Control Methods and Machine Learning Algorithms for Human-Robot Interaction and Assistive Robotics, at the 39th IEEE International Conference on Robotics and Automation ICRA, 23-27 May 2022, Philadelphia, USA.

The following patents based on the thesis were granted by the German Patent Office.

- Deutsches Zentrum für Luft- und Raumfahrt, invented by **S. Bustamante**, D. Leidner, J. Vogel, G. Quere, K. Hagmann , "Verfahren zum roboterunterstützten Ausführen von Aufgaben, Roboter zum Unterstützen eines Benutzers beim Ausführen von Aufgaben und Computerprogrammprodukt", German patent DE102021104883B3, 2021.

- Deutsches Zentrum für Luft- und Raumfahrt, invented by **S. Bustamante**, J. Vogel, F. Stulp, “Verfahren und Roboter zum Ausführen von Aufgaben und Computerprogramm”, German patent DE102022104525B4, 2022.

The following patents based on the thesis are pending evaluation by the German patents authority:

- Deutsches Zentrum für Luft- und Raumfahrt, invented by **S. Bustamante**, I. Rodriguez, D. Leidner, J. Vogel, “Verfahren zum Vorbereiten und Ausführen von Aufgaben mithilfe eines Roboters, Roboter und Computerprogramm”, German patent application DE102022111400A1, 2022.
- Deutsches Zentrum für Luft- und Raumfahrt, invented by **S. Bustamante**, J. Thun, F. Stulp, “Verfahren zum Beantworten von Benutzerfragen, Softwaresystem und Roboter”, German patent application DE102024107724.7, 2024.

#### Publications during my time as a PhD student, but not related to this thesis

I note that the following paper was completed and published during my time as a PhD student, but its content is not related to this thesis in any way. It is related to my master’s thesis:

- **S. Bustamante**, J. Peters, B. Schölkopf, M. Grosse-Wentrup, and V. Jayaram, “ArmSym: A Virtual Human–Robot Interaction Laboratory for Assistive Robotics,” in IEEE Transactions on Human-Machine Systems, vol. 51, no. 6, pp. 568-577, Dec. 2021, doi: 10.1109/THMS.2021.3106865.

# 2

# Fundamentals

While Section 1.4 contained a brief survey of the most directly related works and their research gaps, this chapter provides: (1) A more general theoretical framework for the thesis, aiming not only at the research gaps as in Section 1.4, but also to provide a general survey of shared control in assistive robotics. (2) A detailed technical overview of preceding work (i.e., work from our research institute which precedes and permeates the work on this thesis), which is necessary to understand the subsequent chapters.

The chapter is divided into four sections: I start with an in-depth description of paradigms for shared control support in assistive robotics in Section 2.1. In order to identify my work in comparison to the literature, I categorize the existing paradigms in a taxonomy, drawing parallels with the more general teleoperation literature. Next, I present in Section 2.2 a primer on world models for cognitive robots, with an emphasis on the implementation used in the thesis. After that, in Section 2.3 I present an overview of the Shared Control Templates, the action representation previous to my work. Later, in Section 2.4, I show an overview of the robotic systems I used in the thesis. Finally, in Section 2.5 I show a summary of the research gaps identified in the previous chapter (Chapter 1), which are the base of subsequent thesis chapters.

## 2.1 A taxonomy of shared control support in assistive robotics

Shared control is a very heterogeneous field of research; it receives many definitions and influences a long list of application domains. It is difficult to draw a line with the broader literature body of teleoperation, and with concepts such as shared autonomy. The goal of this subsection is to highlight the works in shared control *specific to assistive robotics*, and to cluster the types of support they provide to users. But I also consider other related works in the literature for shared control in teleoperation, as both fields have had great influence on each other, and share many definitions and concepts.

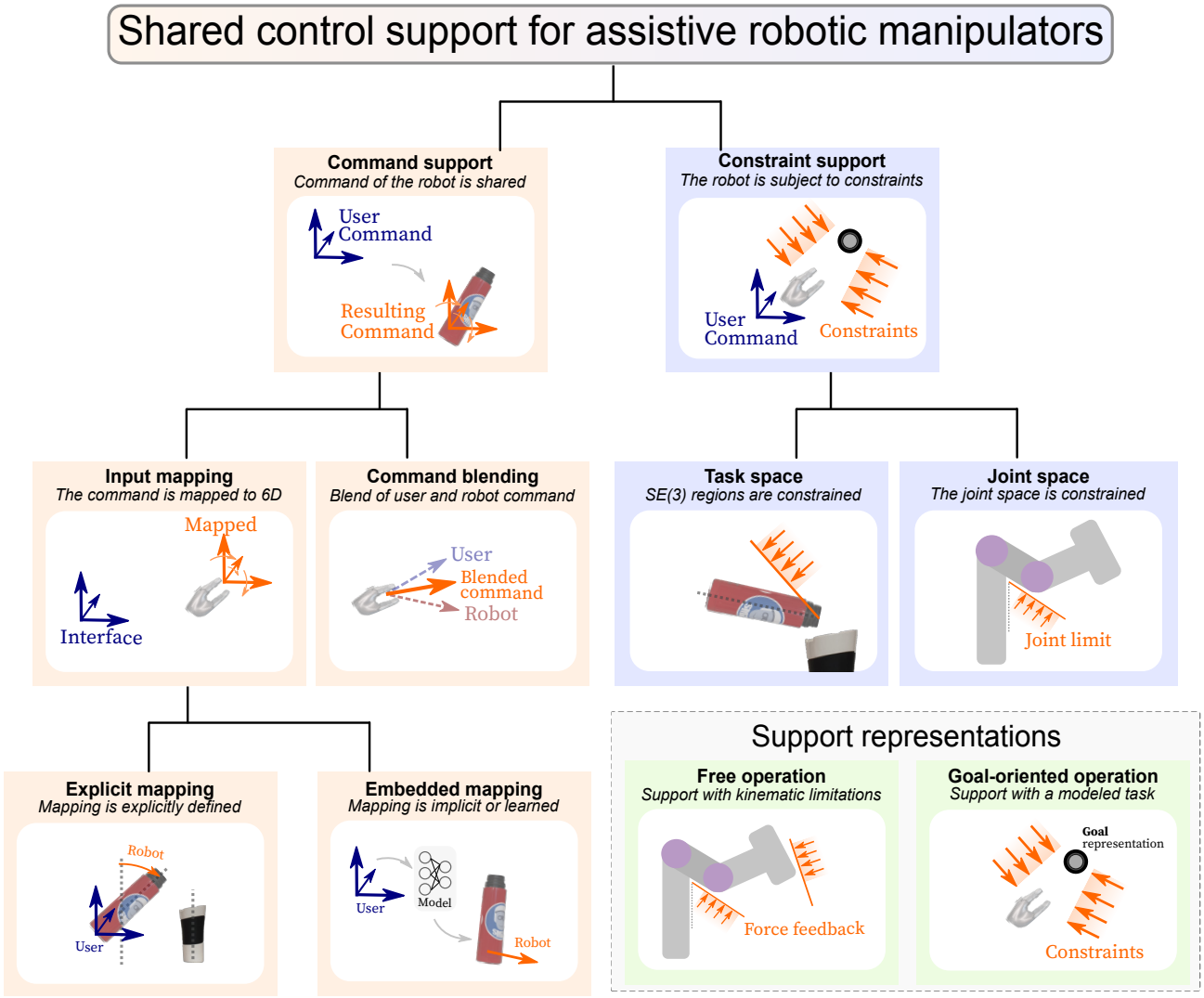



**Figure 2.1:** A brief taxonomy of the types of shared control support for controlling assistive robotic manipulators. The first level divides the two main types of support: on the commanding space (orange branch), by providing support to control the robot EE in six dimensions (three rotational and three translational); and as constraints or virtual fixtures (violet branch) that limit the motion of the EE. In the gray box in the bottom right, I recall a further distinction on the type of support representations in the teleoperation literature (green panel), where the goal-oriented operation is more represented in the assistive robotics body of work. **Credits:** minor design elements were taken from [155, 36] © 2025 Thesis author.

In order to present information in a concise way, and based on Definition 1.1 from the last chapter (i.e.,

> **Shared control** being *"the control modality where robot control variables are jointly controlled by the human and the system [155], either proportionally, split along degrees of freedom of motion, or by means of constraints on the robot body."*, (Def.1.1)

I derive a **taxonomy** of shared control works, illustrated in Fig. 2.1. It classifies works by the *paradigms of support*; in other words, the taxonomy aims to answer the question: how do related works realize the joint control of robot variables? [1] This categorization divides support types as either being an *extrapolation of the user command* (i.e., if the low-dimensional control of the user is mapped to a higher dimension control signal of the robot), or a virtual fixture, i.e., a high-dimensional constraint on the operator [2].

### 2.1.1 Branch 1: command support

This concerns the orange branch. I further distinguish two types of command support in the literature: Commands can be *divided* between the robot and the user, meaning either party takes care of different aspects of the robot movement; or, both robot and user can issue commands, and the output is a *blend* of both.

**Input mapping:** The different degrees of freedom commanded to the robot are **divided** between the user and the robot. A common setup is that the user controls most of the translation degrees of freedom while the robot controls the rotational ones, which are arguably less intuitive to control.

In this category, I highlight a set of shared control methods based on the "expression-graph-based Task Specification Language" (eTaSL [6]). While the robot is controlled with a low-dimensional 3D joystick in [169], for instance, the shared control system implicitly keeps control of the remaining control variables using constraints learned from demonstrations. In the illustration in Fig. 2.2, for example, the user translates the EE within the green tube, while

[1] **Note:** In the context of this section, I assume the terms *robot command*, *robot control*, and *robot policy* all denote the control variables given to the robot, although some communities may find different meanings and understandings.

[2] Note that this distinction is loosely based in Bowyer et. al. [29], which distinguishes between "Guidance" constraints, i.e. "*constraints that encourage the user to move the tool along a specific pathway or toward a specific target*" [29], and "regional" constraints that limit the movement of the robot. However, in the assistive robotics literature, it is common to speak about command guidance (e.g., command blending, input mapping [93, 155]) towards a goal rather than guidance constraints.

the robot controls the orientation. Furthermore, the method reacts dynamically to obstacles (turquoise star) by adapting the support constraints online, based on previous work [193].

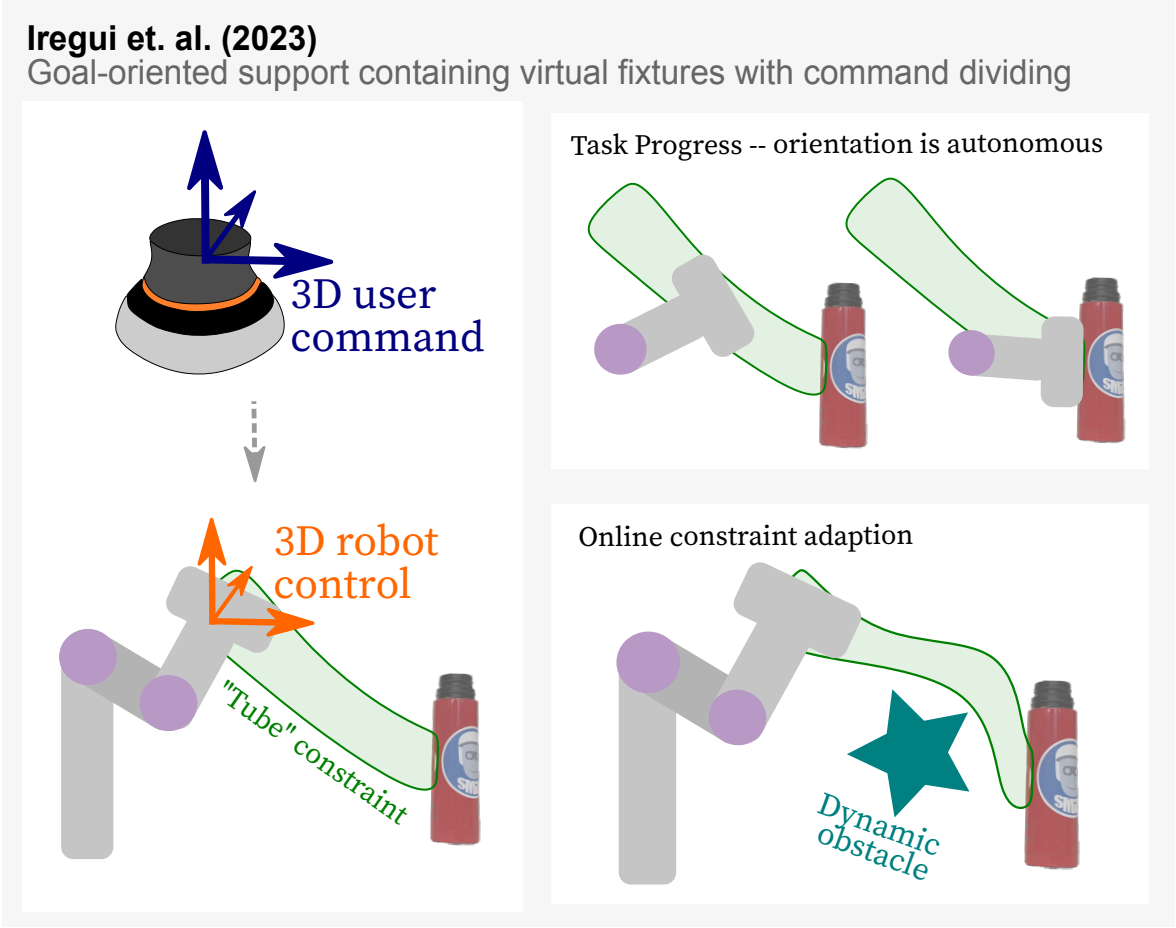



**Figure 2.2:** Visual summary of Iregui et. al.'s shared control method [169]. The user controls an assistive robot on a reaching task via a 3D joystick, in this case a Spacemouse (3Dconnexion GmbH, Munich, Germany). During the task, the robot is subject to an active constraint in the form of a generalized cylinder, colloquially known as a "*tube*", and learned from demonstrations. As the EE progresses within the tube, its orientation is autonomously changed proportionally to the task progress. Finally, the active constraint tube can move and morph in space if there are dynamic constraints, such as the turquoise star. © 2025 Thesis author.

Recent work in this domain has focused on automatic mode switching based on task context, given a language instruction. That way, if the robot is pouring, an LLM can switch the user-controlled DoF automatically from an EE translation to a rotation [189].

In the example in Fig. 2.2 (from Iregui et. al. [169]), the division of degrees of freedom was **explicit**, as the user is in control of the translations, and the robot controls the rotations as a function of the distance to the target. However, this splitting of input commands need not be explicit. Losey and colleagues [124, 123] introduced a method for **embedding** the user command in a deep learning system. The method trains an autoencoder model from human demonstrations, resulting in a low-dimensional latent action $\boldsymbol{z}$ (of the same dimensionality as the user command $\boldsymbol{u}$, for instance from a joystick) and a decoder model $\phi$:

$$\boldsymbol{c} = \phi(\boldsymbol{z}, \boldsymbol{s}) \tag{2.1}$$

that reconstructs robot policies from the latent space, based on a context (state) variable $\boldsymbol{s}$.

On inference time, the user command $\boldsymbol{u}$ is mapped to the latent space using a learned projection $\boldsymbol{z} = f(\boldsymbol{u})$. A robot policy is decoded using Eq. (2.1), i.e.,

$$\boldsymbol{c} = \phi(f(\boldsymbol{u}), \boldsymbol{s}). \tag{2.2}$$

In the simplest case, the mapping $z = f(\boldsymbol{u})$ is simply the identity, i.e., the command is *forwarded* to the latent space as $\boldsymbol{z} = \boldsymbol{u}$. More complex mappings can be built to align the robot behavior to user or experimenter expectations and preferences: for instance, one user may expect moving the joystick down to pour, while others may prefer to move it towards one of the sides [123]. This learning-based approach enables the robot to learn intuitive, context-dependent, and user-friendly command mappings for a variety of daily living tasks.

Related work in the embedding domain for teleoperation [203] models the potential robot action distributions, and employs diffusion models to translate the user actions to that distribution, although this example has not been brought yet to the assistive robotics domain.

**Command blending:** The different degrees of freedom of the robot command are controlled by both the user and the robot, and that control is **blended**. Works in this direction are largely inspired by a teleoperation formalism from pioneering work from Dragan and Srinivasa [2]. Given a user command $\boldsymbol{u}$ and robot command $\boldsymbol{p}$ (for instance, a policy from a motion planner, typically following the inferred goal of the user), the robot control command $\boldsymbol{c}$ is given by the equation:

$$\boldsymbol{c} = (1-\alpha)\boldsymbol{u} + \alpha\boldsymbol{p} \tag{2.3}$$

where $\alpha$ is a parameter that specifies how much control is allocated to the user or the robot.

This formalism influenced many works in the community. Most notably, Muelling's method in Fig. 2.3 enabled two tetraplegic people to control a robot arm on diverse daily living tasks, with a 4DoF invasive brain-machine interface (BMI). [93] The method pre-computed the most likely grasp pose given a scenario, using the so-called "capture envelopes" for snapping users to graspable objects. Instead of directly commanding the robots towards those objects to be grasped, they slowly blended in the robot command on top of the user command, effectively adding the missing degrees of freedom.

A similar method has been integrated in other works. Gopinath et. al. [66], for instance, proposed an autonomy customization module, enabling users to verbally select their desired autonomy levels based on their task preferences. Javdani et. al. use a similar framework in which the robot assists an open-ended manipulation task, while it discovers the goal of the user [90]. More recent works focus on leveraging pre-trained models to reduce the modeling effort required to set up these systems, enabling zero-shot intent detection [19];

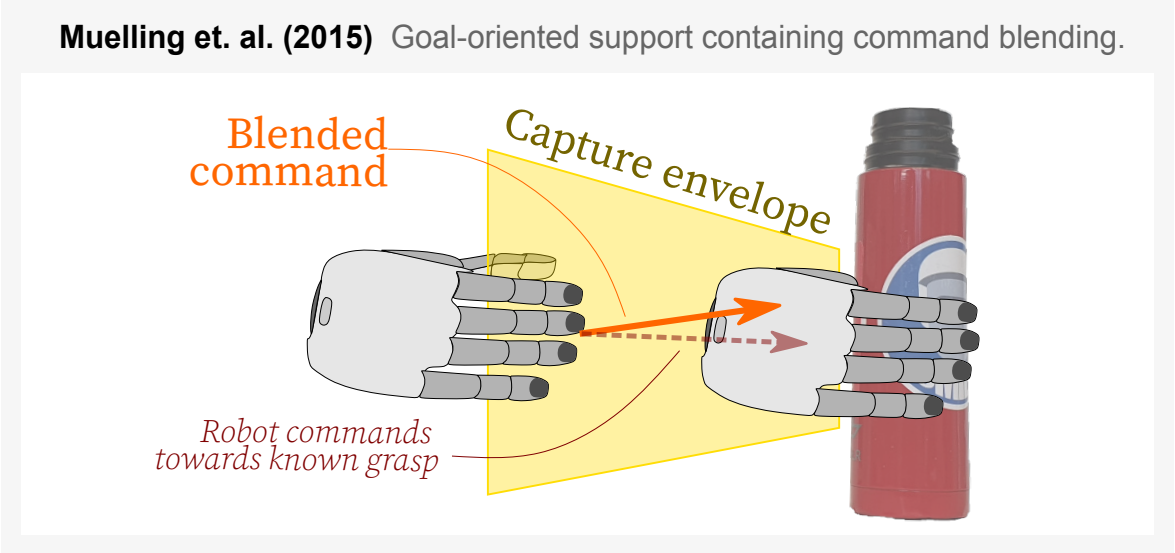



**Figure 2.3:** Visual summary of Muelling et. al.'s shared control method [93]. Objects affording an interaction have a so-called capture envelope, i.e., a region in space where the user is attracted towards the object. The robot issues a command towards a known grasp pose, but before execution, it is blended with the user command as well (not shown). Therefore, the user is able to influence the grasping position, although it is biased towards a known grasp. © 2025 Thesis author.

and on adding methods for information gathering on top of the command blending, including asking questions such as `is <this object> with <this grasp type> your goal?` [204]. Finally, the command blending method extends to interfaces with lower throughput, such as a tongue control system [151] or a non-invasive BMI based on noisy Electroencephalography [104, 202], achieving high success rates in reaching tasks.

The future perspective in command blending work is to incorporate more intelligent and task-aware autonomy, for instance, by leveraging recent advances in foundation models. To give an example, Nakamura et. al. [137] recently developed a so-called *safety filter* for teleoperation. Although not directly using the formalism in Eq. (2.3), this method allows for the robot to override the teleoperator when it deems an action is unsafe (i.e, causing an undesired event), as estimated via an anomaly detector model based on a DINO-WM [209], a world model built on top of the DINOv2 vision foundation model [141].

### 2.1.2 Branch 2: constraint support

This concerns the violet branch. In constraint support paradigms, the degrees of freedom of the robot are controlled by the user, while different areas of the geometric space are restricted by means of virtual fixtures [29], also known as active constraints. Formally, I use

the following definition:

> **Virtual fixtures** (used here interchangeably with "active constraints") *"loosely based on mechanical fixtures"*, are manipulation aids *"which anisotropically limit tool motion, but, as they are enforced in software, they are considerably more flexible with respect to positioning, real- time modification, and feel."* Bowyer et. al. [29] summarizing Rosenberg [168, 167]. (Def.2.1.1)

Virtual fixtures can mean, for example, that the robot EE cannot go anywhere close to the location of a known table, to prevent a collision. I further divide between virtual fixtures in **task space**, for instance, to avoid the environment [180], and virtual fixtures in **joint space**, e.g., to prevent the robot from joint limits [177].

Virtual fixtures are widely used in teleoperation. However, and perhaps surprisingly, not many assistive robotic manipulation works use constraint representations like these. Three examples that implement constraints explicitly for assistive robotics include:

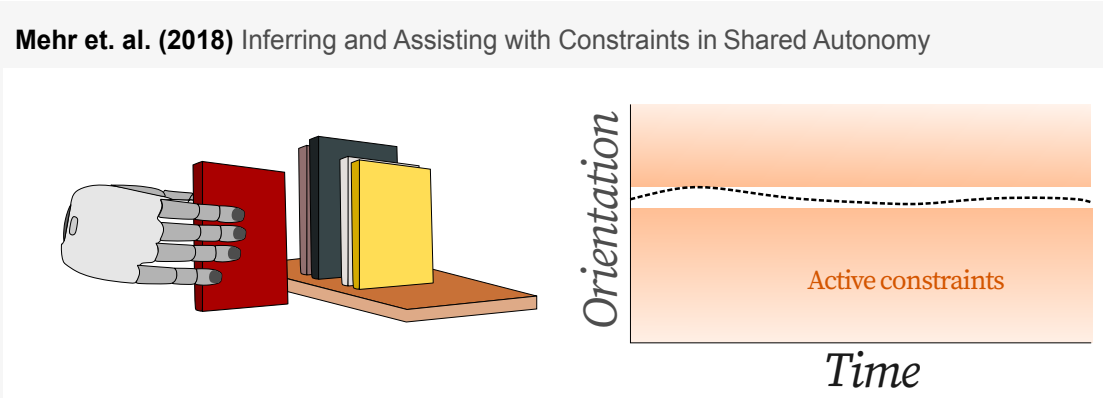



**Figure 2.4:** Visual summary of Mehr et. al.'s shared control method [129]. The method infers constraints from the operator input and assists the user in maintaining them. These constraints can be, for instance, constraints on the orientation of the EE while fetching a book from a shelf. © 2025 Thesis author.

- Mehr and colleagues provided a method that infers constraints the users may attempt to satisfy (such as keeping a water bottle upright, or fixing the EE orientation while retrieving a book from a shelf, shown in Fig. 2.4) [129].
- Iregui et. al. used the eTaSL framework to also provide modular constraints such as generalized cylinders (colloquially known as *tubes*, and shown in green in Fig. 2.2) [169].
- Clanton et. al. (following work on Collinger et. al.) used virtual fixtures on high-dimensional spaces (translational, rotational, or irregular fixture shapes) to enable grasping objects in a BMI [48, 49], following a paradigm for flexible fixture generations that also included tube-like constraints [47].

Other applications exist in healthcare, mainly related to surgical robots with haptic feedback, such as [73, 179].

#### Alternative branch: support representation

The main reason for the gap in constraint support in assistive robotics is the **user interface**. This is due to the fact that virtual fixture works in the literature are highly dependent on haptic feedback [3]. In assistive robotics, however, the interface is so low-dimensional that the focus is rather on achieving a one-directional control direction (and with visual feedback only). Therefore, works in assistive robotics typically represent the user's goal explicitly and rely on task-specific support. While this is often the case in teleoperation too (e.g. [133, 135, 159]), many teleoperation approaches exist where the user is free to operate the space, and constraints provide support against modeled workspace obstacles and kinematic limitations, via force feedback (e.g. [177]). The distinction between these two fundamental representations is also illustrated in the "Support representation" box in the lower right part of the taxonomy in Fig. 2.1 (green panel).

For completeness, and to draw inspiration for our approach (particularly for the feasibility checks in Chapter 4) I also provide an overview of the use of virtual fixtures in the broader context of the teleoperation literature. Two types of constraints exist in the literature:

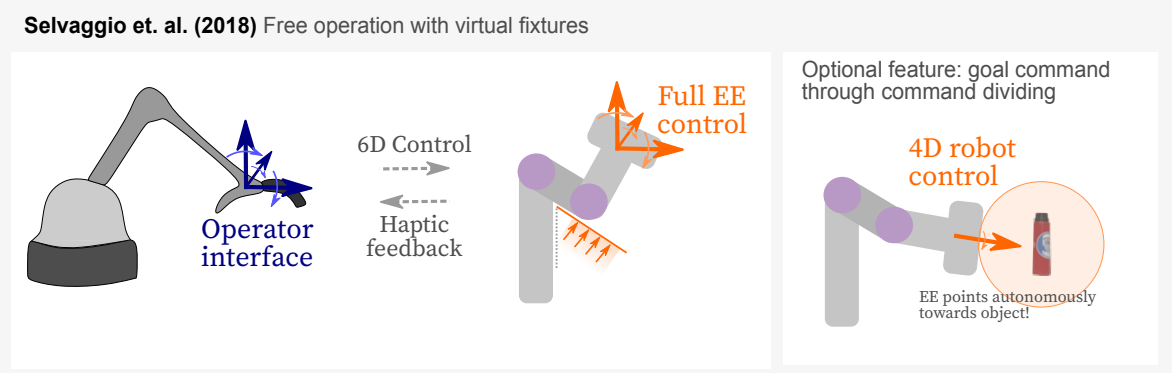



**Figure 2.5:** Visual summary of Selvaggio et. al.'s teleoperation method [177]. The operator uses a 6D interface to control the EE of a robot, and receives haptic feedback from kinematic limitations, such as a joint limit. Left side: normal free operation with virtual fixtures. Right side: an alternative shared control method, where the goal is modeled, and the command is divided. In the particular example, the EE was controlled to always point towards the object, while the user retained 4D control (two dimensions on the sphere around the object, but also on the direction towards the object, as well as the EE roll rotation). This last example is based on previous work from Abi-Farras et. al. [5]. © 2025 Thesis author.

- Fixtures in the robot body in **task space**, i.e. as explicitly blocked regions in SE(3). While these constraints were already used in the early designs introduced by Rosenberg [168, 167], research has focused on making their generation more flexible and

[3] Abbink et. al. even propose haptic shared control as a way for the human to retain final authority over the system, adding traded control capabilities to the gap between teleoperation and assistive robotics.

interactive. Multiple paradigms for adapting virtual fixtures dynamically exist, for instance, dividing the task into multiple sub-tasks with different fixtures (and allowing the operator to switch between them freely) [3], or by providing on-the-fly goal-oriented pedagogical task demonstrators for a space robot [18]. Particularly in unstructured and novel environments, one avenue of research consists of leveraging computer vision approaches to enable constraint-generation on-the-fly, making constraint programming more flexible [178, 180]. In similar cases, adding several vision-based fixtures while retaining flexibility can be realized with mixture-of-experts models based on state variables [133, 135], where virtual fixtures can be stored in libraries and prioritized online [159]. Composable constraints (i.e., representing different features, like specific tools) can also be based on vision, as has been recently demonstrated [150]. Finally, virtual fixtures can also be learned from demonstrations, with representations similar to generalized cylinders [9, 157, 159].

- Constraints in the robot body in **joint space**. The main goal of constraining the joint space is to prevent the robot from falling into kinematic limitations ([39]), such as joint limits or singularities. The robot arm illustrated in Fig. 2.5 uses haptic feedback to enable robust teleoperation, by informing the operator about the command feasibility against the controller's constraints [177]. This is because the controller also contributes to solving kinematic limitations: it not only solves the main teleoperation goal, but also prevents kinematic limitations by applying a secondary objective to the control loop, i.e., it uses the kinematic redundancy of the arm to avoid collisions, singularities, and joint limits. Note that [177] also features an alternative system with *command support*, where the orientation of the EE is controlled by the robot to assist with the grasping while still providing haptic feedback, shown on the right side of Fig. 2.5. Similar approaches have also been explored in other related works, for instance, in applications for nuclear waste sorting [5].

  Finally, to guarantee the safety of such approaches, passivity-based controllers have been developed [176], where one recent line of work has been on enabling safe application of variable stiffness [144].

**Mixing support types:** Do note that these categories are not necessarily mutually exclusive. In fact, virtual fixtures can be easily added on top of frameworks for input mapping and command blending, such as Iregui et. al. in Fig. 2.2. Most notably, our previous work in Shared Control Templates ([155], described in detail in Section 2.3) combines input mapping with virtual fixtures.

## 2.2 Knowledge representation with world models

Research in cognitive robots dates back to the 60's, in the time of Shakey the robot [108]. Shakey, the first cognitive robot, embodied algorithms to *sense* the environment, *plan* actions, and follow the execution of an *act*. This architecture is still the blueprint of modern cognitive robots, with world models being an integral part of it: they model what is sensed, provide information to a planner, and store knowledge necessary to monitor the robot's operation. In this section, I provide an overview of robotic world models, as they are used extensively in the SCTs Section 2.3.

### 2.2.1 Concepts

To operate in the real world, robots need to model it. I follow Herman Bruyninckx's definition:

> **World model:** *"the information the robot has about the world around it, and that needs to be shared between several activities",* Bruyninckx [32], (Def.2.2.1)

where these *activities* are internal components of the robot, like planning or explainability modules. The definition thus implies that the model represents *only* the local physical world in which the robot operates [172], and follows a closed-world assumption: what is not currently known to be true (or cannot be modeled), is false. In short, a world model thus serves as an interface to the other robot modules of what the robot can possibly model.

For world models to support task and motion planning (and, ultimately, the symbolic and geometric methods from the thesis shown in Fig. 1.9), they need to store and provide on demand two types of knowledge:

- Knowledge on the **symbolic state** of the world. This can either represent statements on the state of a single object (so called **unary predicates** like "*the mug is full*", "*the fridge is open*" ) or the joint state of two objects (**binary predicates**, like "*the tea mug is inside the microwave*" or "*the bowl is on the tray*"). Typically, these symbolic predicates are stored in a first-order language, such as the Planning Domain Definition Language (PDDL [64]). In PDDL, predicates can formally be statements of truth. For example, "`opened fridge`" literally means "*it is True that the fridge is opened*", and "`bowl94 on tray13`" means that "*it is True that a bowl instance with id 94 is on the tray with id 13*". These symbolic predicates are later used in Chapter 5 for planning.
- Knowledge on the **geometric state** of the world. This is typically represented using homogeneous transformation matrices (written by convention as $\mathbf{T}^{\text{reference}}_{\text{entity}}$). A world

model contains the pose of known objects and entities with respect to a fixed world origin (called 0, by convention in this work). The translation $\mathbf{T}^0_{\text{obj}}$ is thus given by a 4x4 matrix, consisting of a 3x3 orthogonal rotation matrix $\mathbf{R}^0_{\text{obj}}$ and a 3D translation ($x_{\text{obj}}$, $y_{\text{obj}}$, $z_{\text{obj}}$). More formally, this so-called *transform* is given by:

$$\mathbf{T}^0_{\text{obj}} = \begin{bmatrix} & & & x_{\text{obj}} \\ & \mathbf{R}^0_{\text{obj}} & & y_{\text{obj}} \\ & & & z_{\text{obj}} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

Note that the geometric and symbolic information can be tied together. For instance, a common representation is a tree-based scene graph (e.g [172]), where a parent-child link can represent an object contained within another one. If the bowl is on the tray, moving the tray will move the bowl too. Thus, the binary predicate "`bowl on tray`" represents as well the static relative transform matrix $\mathbf{T}^{\text{tray}}_{\text{bowl}}$. In such a kinematic link, we can compute the pose of the N-th object with respect to the world origin by iteratively multiplying the relative transform of each parent-child relationship:

$$\mathbf{T}^0_N = \prod_{k=1}^{N} \mathbf{T}^{k-1}_k. \tag{2.5}$$

**A note on the terminology:** The term *world model* in this thesis differs from the one used in recent deep learning (DL) literature. Several works refer to a world model as "*a digital twin of the physical world*" [139], a learned model that can predict the future world state (or at least a representation of the robot's sensory data) given its initial state and ongoing motor actions [209, 68, 1]. For instance, one of these models could predict a video of an apple falling from the robot's end-effector after the fingers have been opened. While these models operate on a compressed representation of the world state, and they model the world's physics laws (e.g., to represent the apple falling), they are largely *forward world models*, because their goal is to predict future dynamic states given the current world state. Therefore, they are akin to internal human models that predict the sensory reafference that a specific movement will generate, and that are likely located or orchestrated in the cerebellum [185].

Perhaps a more critical difference is that a robot world model needs to represent a *persistent* world state, according to the definition:

A **persistent world state** is *"one that remains modeled until there is contradictory information, even if the sensory information is temporarily removed"*. (Def.2.2.2)

As an example, if a human walks in front of the robot's camera, the symbolic and geometric knowledge of the bowl being on the tray should be kept in the working memory, unless the robot experiences that the bowl and the tray have been removed.

Finally, in the thesis, I further assume that our world models are deterministic, although there is a large body of work in uncertainty estimation for robotic world model prediction (e.g. [112, 57], to name two examples).

### 2.2.2 Implementation

For my thesis, I use the World State Representation (WSR) world model from Leidner [117], illustrated in Fig. 2.6. The WSR is an object-centric representation that contains a plain list of the objects that the robot knows about being in its vicinity. All static knowledge about the objects is retrieved from the so-called Object Database (ODB), a central repository where the robot's knowledge is contained.

**Polymorphism in the object database:** The ODB relies heavily on the concepts of polymorphism and inheritance. Objects that the robot knows are represented first as functional object classes (by convention, containing a trailing underscore like `_bottle`). Actual instances of the object, like the `ikea_bottle`, inherit properties of `_bottle` like the actions that are possible with bottles (grasping, pouring), while overriding specific properties, like the diameter, the height, or robot-specific grasp sets. Functional object classes also inherit from more abstract classes: `_bottle` inherits some knowledge from `_container`. At the end, all the objects inherit the most abstract properties from the most general object class, `_object`.

**Object instantiation & Informing the world model using perception:** For each new object it needs to interact with, the robot loads one object instance from the ODB into the WSR. This process may be automatic by using a computer vision pipeline, or manual, i.e., in an experiment with known object poses [4].

On EDAN, for example, objects are instantiated based on an object detection and localization pipeline, shown in Fig. 2.6. Objects from the ODB are detected based on a standard bounding-box detector (fine-tuned on ODB objects) and a localization algorithm based on an auto-encoder network [187]. An anchoring algorithm allocates each new object instance with a unique id with a number (e.g. \$14 for the *ikea_thermos* in Fig. 2.6), to ensure the object state and its history are *persistent* (Def 2.2.2), and also to enable multiple instances of the same object in the same world (e.g., a new bottle can be added with the tag *ikea_thermos*\$24 to the world already containing *ikea_thermos*\$14, where all the properties and history of *ikea_thermos*\$14 are kept independently from *ikea_thermos*\$24). The static properties of

[4] A third option is that the world is pre-set by the experimenter, but the robot registers itself to it, locating the objects and their poses later.

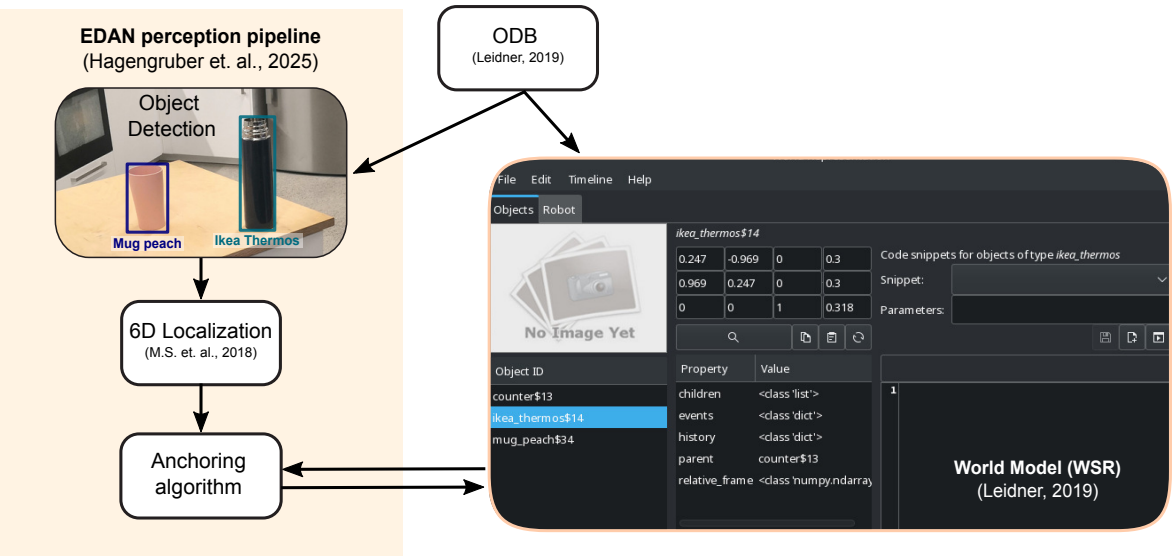



**Figure 2.6:** EDAN's perception and world modeling pipeline. The WSR [117] (**right**) models the robot's beliefs about the world state, containing in this case three objects: a counter, a thermos bottle, and a peach-colored mug. A perception pipeline (**left**) based on the robot sensors can inform the world model. Objects from the ODB [117] can be detected using pre-trained detection models (here, a bounding-box-based model is illustrated. The bounding boxes were drawn manually for the sake of the example), and their pose can be obtained using 6D location methods (such as an augmented autoencoder [188]). An anchoring algorithm keeps track of the robot's movements and computes which located objects are already part of the robot's world model, adding those that are not. **Credits:** This figure has been inspired from the model in [69]. **Note:** this mug also receives the symbol `red_mug` on subsequent chapters.

all WSR objects (e.g., color or mass, where applicable) are automatically loaded from the object database.

But how do we inform the world model about symbolic information of the world? Unary and binary symbolic predicates can also be obtained from perception modules. An efficient method for doing so is SORNET [205], a VLM that can decode object relationships based on a picture of the environment and patches with images of the object classes. DL-SORNET, an instance-centric extension of the SORNET architecture, proposed using segmentation masks from the RGB instead of class image patches, and has been explored in a Master's Thesis in the context of EDAN [109]. Recent developments in VLMs have brought new methods that contribute to the solution of this problem, such as training-free pipelines for state estimation based on pre-coded prompts [99].

**New features during the thesis:** During my thesis, I contributed to two further features on the ODB and the WSR:

- In collaboration with the Rollin' Justin team, we added a tree-based representation of objects, inspired by [171], and used in the daily operation of EDAN and Rollin' Justin. In Fig. 2.6, for example, the Thermos object is the child of the object counter. The frame is thus stored as a relative frame, and its pose with respect to the origin is computed

using Eq. (2.5).

- I added a concept for *object-robot sets*, i.e., ontological descriptions of relations between classes of multiple objects. That way, different combinations of objects (like a specific mug and a robot hand) could have clearer storage of object properties. The sets are described in more detail in section 5.2.4.

## 2.3 Shared Control Templates

The Shared Control Templates (SCTs, introduced by Quere et. al. in 2020 [155], formalized further in Quere's PhD thesis in [153]) are our main action representation for Shared Control, and the base of my work in this thesis.

An SCT is, essentially, a template for providing shared control in a specific task defined for functional classes (e.g., pouring from a `_bottle` into a `_container`, illustrated in Fig. 2.7). In a given context, defined by the WSR, an SCT is instantiated with specific objects, like an `ikea_bottle` in the hand of the robot pouring into the `mug_peach` on the table.

In this subsection, I describe the elements and assumptions of the SCT framework.

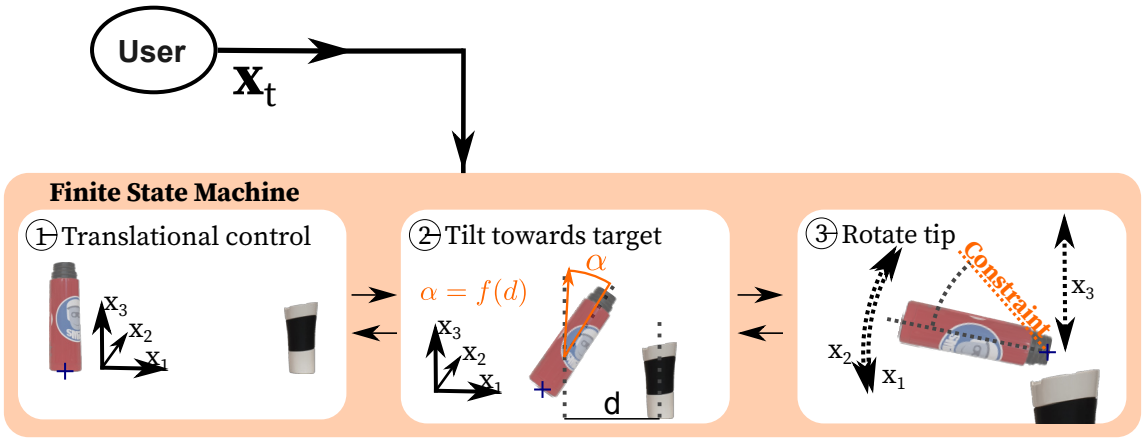



**Figure 2.7:** A schematic an SCT. We show a Shared Control Template for pouring from a `_bottle` into a `_container`. This includes a Finite State Machine. The user (top left) can issue commands $x \in \mathfrak{R}^{\text{dim}}$ to move the robot. For the case where dim = 3, I show the components $x_1, x_2, x_3$, and how they are mapped to motions of the bottle in different states, described in detail in the text. **Credits:** First presented in [155], mainly featured in [36]. Modified for this thesis. © 2021 IEEE.

### 2.3.1 SCT input

The user controls the system via an *input command* $\boldsymbol{x}_t$, with $\boldsymbol{x} \in \mathfrak{R}^{\text{dim}}$, where dim is the dimension of the input interface of the user.

In our experiments, we commonly use dim = 3, as we have found it gives a balance between tasks being intuitive and controllable enough, and the interfaces available for our

users. An argument for this is that at all times, it allows for covering either all translational or all rotational EE motions. However, without shared control, operating the robot would require mode switching between EE translations, EE rotations, grasping, and wheelchair motions.

For the remainder of the thesis, I assume that $\boldsymbol{x}_t \in \Re^3$.

**Input device:** For all my experiments in the thesis, I used a 3D joystick with additional binary buttons for task-related variables. In Vogel et. al. [194] and Hagengruber et. al. [69, 70], my colleagues explore different interfaces like surface electromyography and combinations of electromyography and joysticks. However, these are not explored in my thesis.

### 2.3.2 Finite State Machines (FSMs)

The main element of an SCT is a Finite State Machine (FSM). FSMs divide a given task into *states*, and further define *transitions* between them. As shown in Fig. 2.7, the task for pouring has different states where the user is supported by the robot in different ways.

Transitions between states happen after certain pre-defined *events* in the task, like when the object in hand reaches a certain pose, position, or orientation. In Fig. 2.7, for instance, the transition between state "Tilt towards target" and "Rotate tip" happens when the distance between the objects is less than a threshold, which is pre-defined by the skill designer.

**Other variables:** In general, any variable of interest within the task can trigger a transition: measured forces, distances between any two arbitrary frames of interest, and the angle between two frames are three examples of transition variables [5].

I describe next how each state provides customized support to the user.

### 2.3.3 Support within a state: Input Mappings and Active Constraints

Each state in WSR defines two sources of support, each related to the taxonomy in Fig. 2.1:

- Input mappings (IMs) represent **command dividing**, i.e., they define how the low-dimensional user inputs $\boldsymbol{x}_t$ are mapped to relevant EE motions within a state.
- Active Constraints (ACs) represent **virtual fixtures** that restrict the end-effector from going into certain regions of the workspace.

**Input mappings:** The three components of the user command $\boldsymbol{x}_t \in \Re^3$ are mapped to the robot EE motions. In other words, every state has its individual support motions, provided

[5] The robot and related objects are modeled as rigid bodies

by the skill designer, such that they are as intuitive as possible. *Example:* Consider Fig. 2.7: In state 1 (*translational control*), the bottle has not started rotating yet, so the components of the user command are simply mapped to the translation of the grasped bottle. Thus, the user translates the EE and the bottle. But two states later, in *rotate tip*, the commands of the user are divided: a vertical dimension $x_3$ takes care of the vertical translation, whereas two horizontal components $x_1$ and $x_2$ map to rotations of the bottle tip.

As previously mentioned, IMs implement command dividing. In *Tilt towards target*, in state two, the user is still in charge of the EE translations, but the robot provides the EE rotations, applied to the bottle frame. The angle of the bottle is a function of the distance between containers, which effectively means that the closer the user gets to the white container, the closer the bottle tip is to start pouring.

Finally, I note here an important property of SCTs: the implementation of IMs is such that if the user is not issuing any commands, the robot does not move. The user is in full control of the directions (and, implicitly, of the speed) of the robot motions.

**Active constraints:** In SCTs, ACs are represented as either linear (e.g., stay below a line) or volumetric primitive constraints (stay within a sphere). The framework employs so-called proxy methods [29] to realize these constraints kinematically.

To formalize this, let $\mathbf{T}_{\text{t-unconstrained}}$ be the pose resulting from applying the state input mappings to the user command $\boldsymbol{x_t}$. If $\mathbf{T}_{\text{t-unconstrained}}$ violates an AC definition (e.g., it is above the line, or out of the sphere), it is *projected* into an EE pose $\mathbf{T}_{\text{t-constrained}}$, the nearest pose that does not violate the constraint. This is done by means of a projection function:

$$\begin{aligned} proj\colon SE(3) &\rightarrow SE(3) \\ \mathbf{T}_{\text{t-unconstrained}} &\mapsto \mathbf{T}_{\text{t-constrained}}. \end{aligned} \tag{2.6}$$

To make a graphical example in Fig. 2.8, an AC may define height limits to the EE while grasping a bottle. If the pose goes over the height threshold, it would be projected into the nearest point that respects that limit, which is the threshold itself. To give another example: if the SCT requires the EE to always point towards an object to be grasped, its Euler angles (e.g., the yaw) can have a threshold constraint as well.

Finally, the output of the ACs is a final pose $\mathbf{T}_{\text{t-constrained}}$. From now on, I drop the reference to the ACs and refer to this pose[6] simply as $\mathbf{T}_{\text{t}}$.

### 2.3.4 SCT output

The SCT IM and ACs combination outputs a pose $\mathbf{T}_{\text{t}}$. We take two further processing steps:

[6]It is worth noting that SCTs can apply multiple ACs in a specific state, by iteratively applying projections. In that case, the output pose is the result of the last projection.

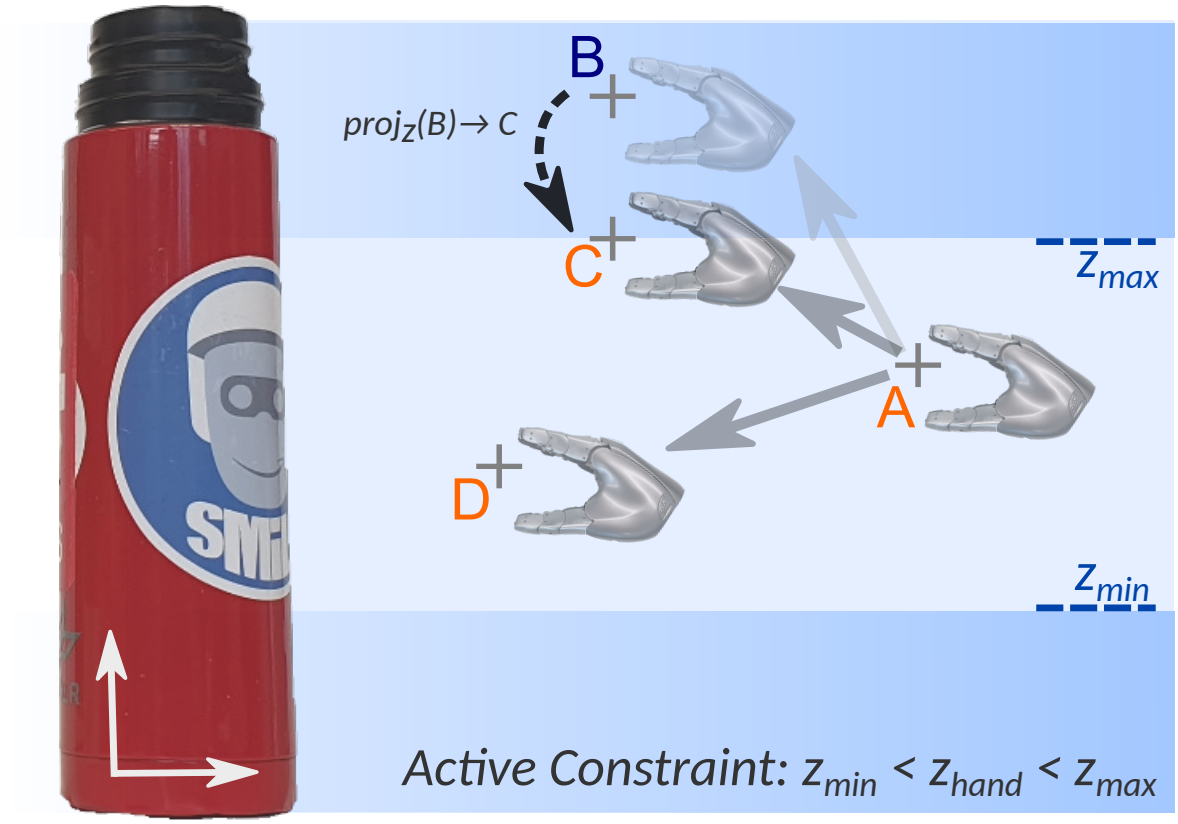



**Figure 2.8:** An active constraint for grasping a bottle. The AC defines height limits $z_{\text{min}}$ and $z_{\text{max}}$ with respect to the bottle's coordinate axis. Let A be the pose of the EE before a command. If the command (after the IMs) brings the robot to pose B, it will violate the constraint. The EE is thus projected to the nearest point of B that does not violate the constraint, which in this case is C. Thus, even if a command repeatedly pushes the EE in the upper direction, the robot EE will never be over $z_{\text{max}}$. Finally, if instead of B, the EE had moved to pose D, the projection would not have been necessary, as the constraint is respected. © 2025 Thesis author.

First, $\mathbf{T}_t$ is sent as the target of a spherical linear interpolator [199], which ensures that changes to the EE are resolved smoothly (i.e, without sudden jumps) and with velocity limits for safety.

Second, a Cartesian impedance controller for the arm guarantees that the EE is compliant during the motion. On EDAN, this includes a whole-body controller to enable simultaneous motions of the arm (8DoF) and the wheelchair (2DoF). That way, if a goal pose is out of reach for the arm, the wheelchair can move to give it more room.

### 2.3.5 SCT activation via the world model and action graphs

In nominal operation of EDAN, no SCT is activated, i.e., the user is in control of the EE's translations (and of rotations, via mode switching [78]).

SCTs can be *activated* to start support of the task, using two interfaces:

- by using a UI, such as a tablet interface; or

- by activating SCTs given by the affordances of objects in the proximity of the EE. Simultaneously, the distances to different objects (as represented in the WSR world model) are monitored. If the distance to an object goes below a threshold, the first state of that SCT is activated.

#### Action filtering with PDDL and action graphs

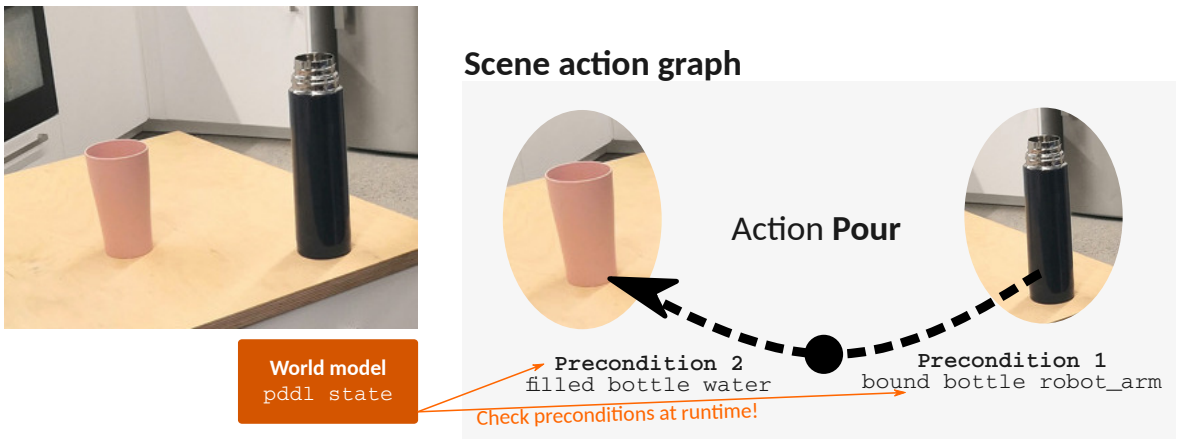



**Figure 2.9:** Overview of the action graph method for filtering actions, based on the Master's Thesis of Mohammed Behery [17]. Given a set of objects in the world state, actions related to objects are parsed, regardless of the state of those objects. The action graph builds actions as *paths* between the relevant objects. Depicted is the path for the action Pour, which has two preconditions: `bound bottle robot_arm` (the bottle has to be grasped) and `filled bottle water` (the bottle has to contain water). During runtime, the algorithm monitors the evolution of those two predicates in the world model. If they become `True`, the action Pour would be available for the user to start.

But how do we know which SCTs can be executed at a given time? SCTs contain descriptions of PDDL preconditions before execution, which allows the robot to filter which SCT actions are possible in a given context. In order to do this, a so-called *action graph algorithm* is used to understand which actions are afforded by the objects currently in the world, and which ones of those can be executed with the state of those objects. This action graph algorithm was first presented in the CARE framework [17] as shown in Fig. 2.9, and later ported to SCTs in [196].

**Short summary of action filtering:** all actions that the world objects allow are stored in a graph, a portion of which is shown in Fig. 2.9. That way, pouring from a bottle will only be offered:

- if the objects that afford the action `pour` (an instance of a `\_bottle` and an instance of a `\_container`) are present in the world, the action is added to the action graph as a path between the objects. Each part of this path represents a precondition in PDDL.

- if all the preconditions in the action graph are `True`. The algorithm monitors the state of those preconditions in the world model, and updates their state (`True` or `False`) in the

action graph. The action is filtered as ***available*** only if all preconditions are satisfied. If at least one is not true, then it is filtered out.

I note that SCT states also contain predicates for effects, in order to update the world model after an action happens.

### 2.3.6 Ease of design: learning SCTs

To ease the tedious work of skill designers, SCTs can be learned from user demonstrations. Three frameworks exist:

- A semi-automated method that performs segmentation on demonstrated trajectories, and then matches them with a pre-existing model library of parametric constraints [154].
- A collection of methods for learning IMs and transitions using linear models and vector fields [10].
- A method with a human in the loop to learn ACs, based on probabilistic movement primitives [157].

### 2.3.7 Limitations of the SCTs

Our team has used the SCTs for several years. Over that period, we have identified some limitations. I list them here and provide some methodological improvements for them in the remainder of the thesis:

1. As SCTs are based entirely on a combination of command dividing with virtual fixtures (Fig. 2.1), they do not provide a natural way for allocating autonomy to the robot, which would be the case with command blending. An autonomy switch would thus require re-planning, which entails time and causes difficulties in representation alignment.
2. SCTs do not directly model obstacles and robot kinematics while computing IMs and ACs. Therefore, users can very easily fall into kinematic limitations: colliding with an object or the robot body itself, falling into low-manipulability situations due to joint limits, or finding zones with singularities.
3. As SCTs are activated based on distance thresholds only, they do not take into account complex kinematic situations or obstacles during activations. For example, SCTs may activate a task to grasp an object behind the wall of a shelf, assuming the robot can simply traverse the barrier.

4. Finally, ACs are represented as projection functions given by Eq. (2.6), but this is an implicit representation, not explicit. This means that they cannot be used as-is in motion planning.

## 2.4 Robots

The experiments in this thesis ran on two robots, shown in Fig. 2.10:

- The wheelchair-based mobile manipulator **EDAN** consists of a 8 degree-of-freedom robot arm with attached to the base of a wheelchair [196]. During the thesis, I use two variations of the EDAN EE: an anthropomorphic hand with five fingers (mostly for the early experiments in Chapter 3), and a three-finger flexible hand (for most of the remainder of the manuscript).
- The humanoid service robot **Rollin' Justin**, which consists of a mobile platform with a robotic torso, two 7-degree-of-freedom robot arms, an anthropomorphic hand, and an actuated head [28]. For shared control experiments, only Justin's right arm was used. Justin's experiments were featured only in Chapter 3.

Both robots ran the same cognitive software stack that included the ODB, the WSR, and the SCTs (although with different perception systems), as well as the underlying control setup including Cartesian impedance control [196]. Only the mobile base of EDAN was used; i.e., on Justin, whole-body control was not utilized.

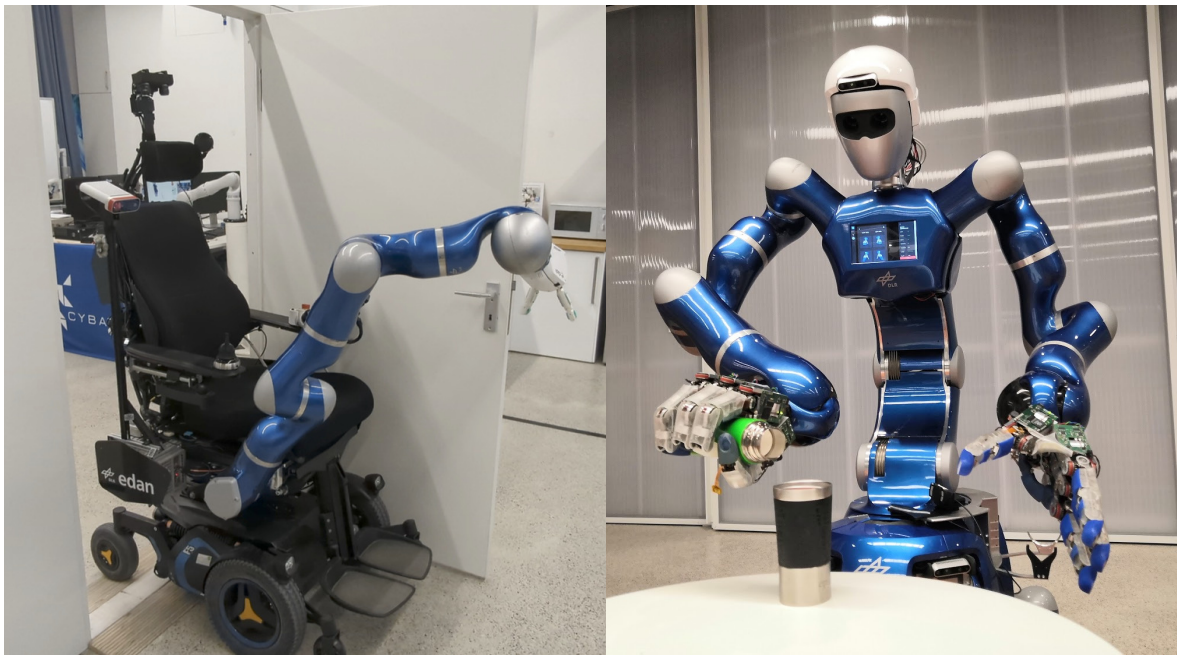



**Figure 2.10:** Robots used in this thesis, the wheelchair-based mobile manipulator EDAN (left), and the humanoid assistant Rollin' Justin (right).

**Digital twin:** Some experiments of EDAN were carried out in simulation, in Chapter 4 and Chapter 6. This simulation was a digital twin of the robot, including the ODB and the WSR stack.

## 2.5 Outlook of the contributions given the taxonomy

To finalize this chapter, allow me to summarize the research gaps I identified in the introductory chapter, in light of the theoretical framework obtained in the current chapter:

1. Representations with input mappings and active constraints, such as the SCTs, can provide shared control, but they do not naturally contain a representation of autonomy behavior. Furthermore, representations for blended commands (1) lack a representation in full shared control, as support *always* comes from an autonomy mode; and (2) need to always integrate an autonomy representation (e.g., via a motion planner or capture envelopes).

   If a shared control representation is used (such as the SCTs), integrating transitions between levels of autonomy results in architectures with two parallel representations, and the need to re-plan every time there is an autonomy switch. Therefore, there is a need for a method integrating autonomy (and *seamlessly*!) with shared control representations → **chapter3**.

2. Methods for haptic feedback via high-dimensional interfaces and virtual fixtures can handle kinematic limitations well. However, there is a need for methods that enable low-dimensional interfaces to handle them as well. One promising avenue is feasibility checks before starting executions of shared control tasks. → **chapter4**.

3. There is a need for goal guidance in shared control actions. Can we provide methods that reason about the initial state of the environment, and provide a plan towards a goal state? → **chapter5**.

4. There is a need for explainability in robotics, to foster system trust. Can the robot provide explanations, in rich natural language, about its internal beliefs, and in a contrastive way? → **chapter6**.

# 3

# Shared control with integrated autonomy

*Flexibility is a key element in our concept of cognitive shared control. 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.*

*This chapter extends the SCT approach with an autonomy module that enables users to seamlessly switch between shared control and supervised autonomy, i.e., at any time during task execution. The chapter summarizes the theoretical foundations and highlights experiments on EDAN and Rollin' Justin.*

***This chapter originates from a paper for the IEEE Robotics and Automation Letters, and I include some literal text portions and figures. However, it has been reworked to fit as a chapter within the thesis concept, and also features a few extensions such as Section 3.2.3 and some figures. See: [36]. Copyright note in the footnote*** [1]**.**

Users of assistive robots can have different engagement interests depending on the time of the day and factors such as workload, system trust, or tiredness. Therefore, a cognitive shared control framework should enable them to seamlessly allocate control to themselves or the robot on a given task, and at their best convenience. This chapter introduces *Shared Control with Integrated Autonomy (SCIA)*, an extension to the SCT framework that provides users with the freedom to allocate autonomy within shared control tasks.

As SCIA uses *only* the SCTs to support different levels of autonomy, transitions between shared control and supervised autonomy are seamless. Here, seamlessness means that the user can switch to supervised autonomy in any task at any point during execution, and also switch back. The novelty of this approach lies in the fact that, traditionally, switching between distinct levels of autonomy requires using different action representations, necessitating the

[1] Whenever parts of the manuscript are taken literally: © 2021 IEEE. Reprinted, with permission from the senior author, from S. Bustamante, G. Quere, K. Hagmann, X. Wu, P. Schmaus, J. Vogel, F. Stulp, D. Leidner, Toward Seamless Transitions Between Shared Control and Supervised Autonomy in Robotic Assistance, IEEE Robotics and Automation Letters, March 2021.

robot to plan anew with every switch. Our approach addresses this by implementing the autonomy switch in the same space where the user provides input commands, such as the 3D joystick, and using the SCTs to provide task support (including the state machine with the transitions, the input mapping, and the active constraints). In other words, the autonomous module, known as the automaton, is provided with the same shared control interface as the user.

This chapter is organized as follows. I first begin by introducing the chapter assumptions in Section 3.1. Then, I describe the method in Section 3.2, including anecdotal evidence of an application of SCIA in Section 3.2.3. After that, I show a set of experiments in the wheelchair-based mobile manipulator EDAN and the humanoid assistant Rollin' Justin Section 3.3. Finally, I present a conclusion, with an outlook to subsequent chapters in Section 3.4.

## 3.1 Assumptions

In Section 2.3 I first introduced the SCTs. The building blocks are the combinations of IMs and ACs, which take a user command as input and output a pose. There is an important assumption about this framework:

**For IMs and ACs to support the user, they must describe task regions.** IMs and ACs define[2] a region of allowable end-effector poses on a given state. Let this region be $\mathcal{S}_s$, where $\mathcal{S} \subseteq SE(3)$ and $s$ is a state of the FSM. In other words, each state defines a different *task region*, a concept that I will formalize in Chapter 4 using the work of Berenson and colleagues [21].

But what properties do these task regions have? Allow me to have a closer look: while in state $s$ at an end-effector pose $\mathbf{T}_{t-1}$, given a user command $\boldsymbol{x}_t$, the Shared Control Template outputs a new pose $\mathbf{T}_t$ in Cartesian space,

$$\begin{aligned} \varphi &: \mathrm{SE}(3) \times \Re^3 \rightarrow \mathrm{SE}(3) \\ \mathbf{T}_t &= \varphi_s(\boldsymbol{x}_t, \mathbf{T}_{t-1}), \end{aligned} \tag{3.1}$$

where $\varphi : (\Re^3 \times SE(3)) \rightarrow SE(3)$ is the combination of IMs and ACs. I further introduce a **fundamental SCT property**: in order to guarantee a continuous and safe shared control task, for every possible user command, the outcome pose of $\varphi$ must remain in the task region. More formally, $\forall x_t$ [36],

$$\text{if } \mathbf{T}_{t-1} \in \mathcal{S}_s, \rightarrow \varphi_s(\boldsymbol{x}_t, \mathbf{T}_{t-1}) \in \mathcal{S}_s, \tag{3.2}$$

if a transition does not happen after command $\boldsymbol{x}_t$; or,

[2] Alas, this is only an implicit definition, as the region is not made explicit in the SCT. This is an issue of the method which I discuss further in Chapter 4.

$$\text{if } \mathbf{T}_{t-1} \in \mathcal{S}_s, \rightarrow \varphi_s(\boldsymbol{x_t}, \mathbf{T}_{t-1}) \in \mathcal{S}_{s+1}, \tag{3.3}$$

if a transition from state $s$ to state $s+1$ does happen after command $\boldsymbol{x_t}$, where I note that only one transition can happen at the time.

This property describes that the robot will stay in a task region regardless of any command from the user; and that, on entering a new state, the robot is already inside the task region of this new state. As an example, if the robot is in the state "Tilt towards target" in Fig. 3.1, a user command will not cause the end effector to suddenly travel half a meter in a certain direction, or reorient the hand causing the liquid to spill [3]. And, more critically: when the transition from "Tilt towards target" to "Rotate tip" happens, the bottle tip is assumed to be already positioned, such that a small tip rotation will cause the pouring to start, without the need for a large reconfiguration. This property reflects the principle of Shared Control: through a low-dimensional user input, the robot actions *always stay* on a Shared Control task region, providing safety.

But of course, enforcing this property is difficult, and it is the responsibility of SCT designers to verify that it is the case[4]. One pitfall of the original SCT formulation is precisely that it does not account for gaps in transitions. For example, going from one state to the other with conflicting constraints can cause large jumps in task space. While the ancillary representations introduced in Chapter 4 and Chapter 5 help mitigate this problem, our methods (in the remainder of the thesis) assume that the skills are designed in such a way that the fundamental SCT property in Eqs. (3.2) and (3.3) is fulfilled. This is the case of all of our in-house SCTs, corresponding to a wide variety of daily living tasks.

## 3.2 Shared Control with Integrated Autonomy

The key idea for enabling seamless transitions between shared control and autonomous modes is to implement autonomy within an SCT. Instead of using a different action representation for autonomy, SCIA reuses the SCT, and uses a so-called *automaton* module that can provide input commands to the SCT, as illustrated in Fig. Fig. 3.1. During a task in SCIA the robot always stays within an SCT, and the input can be switched forth and back between the user and the automaton. This means that the SCT is agnostic to whether a command $\boldsymbol{x}$ comes from either of them, and applies the same state transitions, input mappings, active constraints and whole-body control irrespectively.

[3] unless, of course, the task region $\mathcal{S}_s$ of that state is described that way.

[4] One measure we take is clipping the command to a maximum threshold, to prevent very large jumps. SCTs have built-in velocity limits as well.

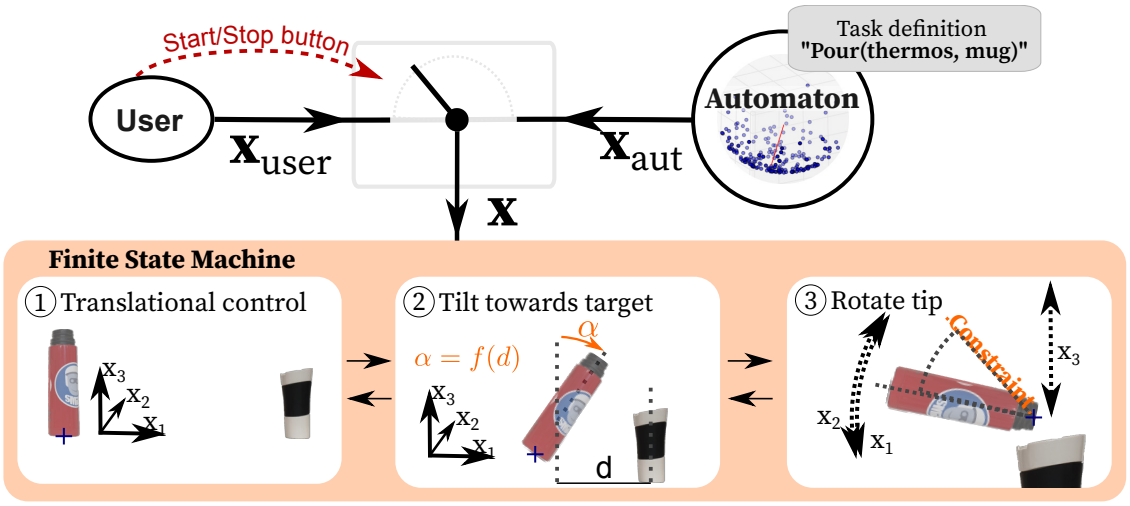



**Figure 3.1:** A schematic of our Shared Control pipeline, and how SCIA connects to the existing infrastructure in Fig. 2.7. SCIA is based on the original SCTs, but adds an automaton algorithm (depicted in the top right) whose goal is to issue commands to finish the task. In normal operation the automaton is deactivated, but can be activated (or deactivated again) at a button press. **Credits:** This figure was taken and slightly modified from [36]. © 2021 IEEE.

**End-to-end examples:** Figure Fig. 3.2 shows an example of the autonomy switch on a grasping task. Other examples of the switching while on a task sequence are publicly available in a YouTube video [42]; an example of an autonomy switch happens at around 2:06 into the video.

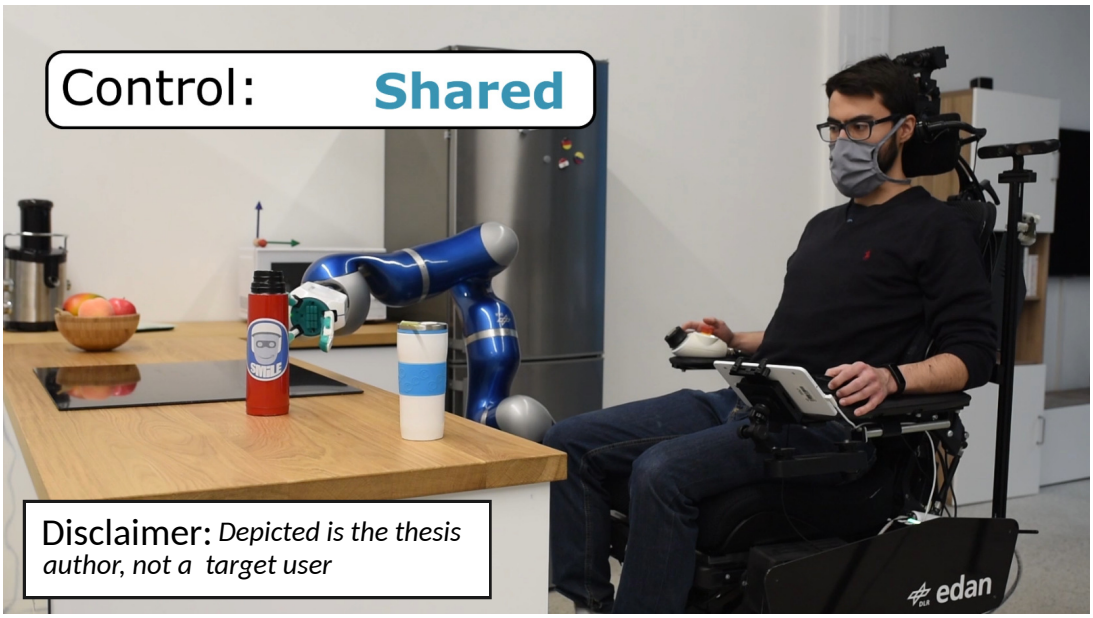
### 3.2.1 Autonomous execution within a state

Given a set of planned states with their different transitions, and IM/ACs in each state, the main question for autonomous execution of the task is: which sequence of input commands should be generated to pass through the different states and complete the task? Expressed from the perspective of the current state: which input commands enable the transition to the next state in the FSM? We describe next how the automaton generates these commands.

#### Loss function

The automaton's immediate aim is to arrive at an end-effector pose where a transition to the next state $s+1$ will be valid. We formalize this with a distance metric, $\mathcal{D} \in \Re$, that relates the current state with the event of a transition. Two examples are shown in Fig. 3.3. In *a*), $\mathcal{D}$ corresponds to the Euclidean distance $\|\boldsymbol{D}\|$ between the mug and the bottle. As the bottle (in hand) approaches the goal, the transition will happen when it reaches a threshold, denoted by a dashed line. Similarly, in *b*), $\mathcal{D}$ refers to the rotation angle $\theta$ of the thermos. More interestingly, in both cases, this metric allows us to evaluate three different bottle poses, giving the automaton clear knowledge of when it is getting close to the next target state.

We define this metric as a loss,




(**A**) The user starts grasping in shared control, and selects the grasping angle:




(**B**) The user presses a button... (**C**) and the robot switches to supervised autonomy.

**Figure 3.2:** Example of autonomy trigger. The user starts a grasping task in shared control, selects a grasping angle (**A**), but then decides to switch autonomy and presses a button that activates the automaton (**B**), switching the robot to finish the task in supervised autonomy (**C**). The entire time, the robot is inside the "grasp bottle from side" SCT.

$$\mathcal{L}_{\text{kinematic}}(\boldsymbol{x}_t) = \mathcal{D}_s^{s+1}(\mathbf{T}_{\text{t}})$$
$$\mathcal{L}_{\text{kinematic}}(\boldsymbol{x}_t) = \mathcal{D}_s^{s+1}(\varphi_s(\boldsymbol{x}_t, \mathbf{T}_{\text{t}-1})), \tag{3.4}$$

in order to evaluate which poses $\mathbf{T}$ can bring the robot closer to the goal. Note therefore that $\mathcal{L}$ will be influenced by the automaton's choice on input space $\boldsymbol{x}$, given (3.1).

*Metric choice:* Recall that this transition metric can be any measurable scalar variable that relates any two frames of interest in the workspace[5]. To name a few examples, the SCT programmer can define a transition as a threshold in (a) the vertical distance between the

[5]This is similar to some related work in constraint-based action representations [12].

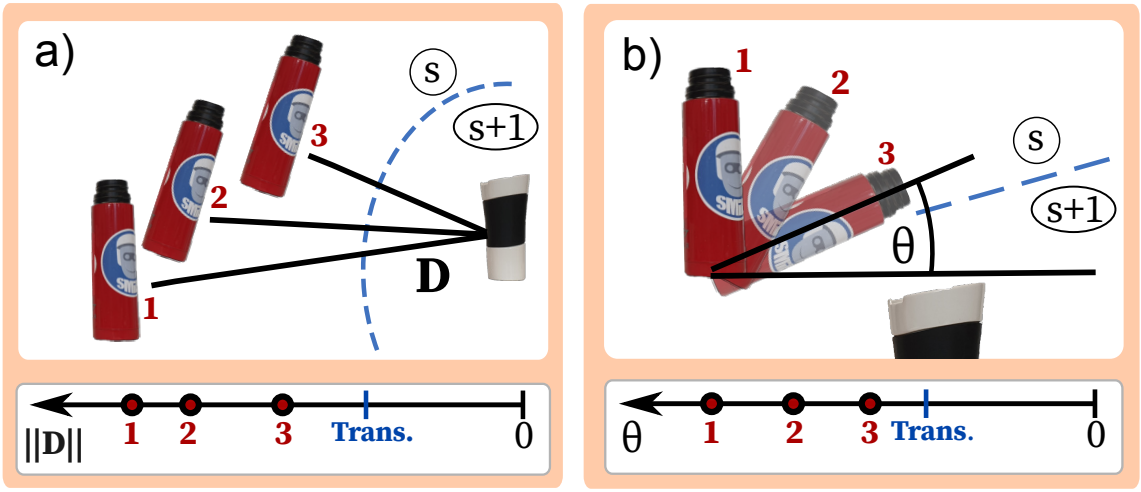



**Figure 3.3:** Two examples of our framework's loss function. In *a*), the transition from state $s$ to $s+1$ depends on the Euclidean distance $\|D\| \in \Re^1$ between the mug and the bottle. Using this metric, we can compare three different bottle positions in the real line, with bottle 3 the closest to the transition and bottle 1 the farthest. In a different example, *b*), the transition does not depend on the Euclidean distance, but rather on the angle $\theta$ of the bottle. Projecting the angle into the real line, we see that here bottle 3 is the closest to the transition, and bottle 1 is the farthest. **Credits:** This figure was taken from [36]. © 2021 IEEE.

hand of the robot and the handle of a drawer; (b) the Euclidean distance between the tip of a spoon and the base of a pot; or (c) an Euler angle between the fingers and a faucet.

**Wrenches:** Transitions in some states can also be based ad-hoc on end-effector wrenches, instead of kinematic goals. For instance, the SCT skill *release bottle* in Fig. 3.4 expects a vertical force (in orange) when the object hits the table, and only then proceeds with subsequent states. Therefore, the goal of the user in those states is to make the end-effector move in the direction that will produce a reaction with the expected force, i.e., moving down (shown in yellow).

Whenever a wrench transition is required, the task of the automaton is to produce commands that move the EE as close as possible to this desired direction. More formally, let the vector $\boldsymbol{v}_{\text{des}}$ be the desired direction of motion for a transition from state $s$ to $s+1$, and $\boldsymbol{v}_t$ be the direction in which the robot moves after a command $\boldsymbol{x}_t$ (i.e., $\boldsymbol{v}_{\text{des}}$ is the vector from $\mathbf{T}_{\text{t}-1}$ to $\mathbf{T}_{\text{t}}$), all shown in Fig. 3.4. The automaton goal is to minimize a loss defined as the angle $\gamma$ between them, i.e.,

$$\mathcal{L}_{\text{dinamic}}(\boldsymbol{x}_t) = \gamma(\boldsymbol{v}_t, \boldsymbol{v}_{\text{des}}), \tag{3.5}$$

where I recall that $\boldsymbol{v}_t$ is a function of the command $\boldsymbol{x}_t$, as it is the vector between $\mathbf{T}_{\text{t}-1}$ and $\mathbf{T}_{\text{t}} = \varphi_s(\boldsymbol{x}_t, \mathbf{T}_{\text{t}-1})$.

Eq. (3.4) is used for all transitions $s \rightarrow s+1$ based on kinematic variables, whereas Eq. (3.5) is used in those based on wrenches.

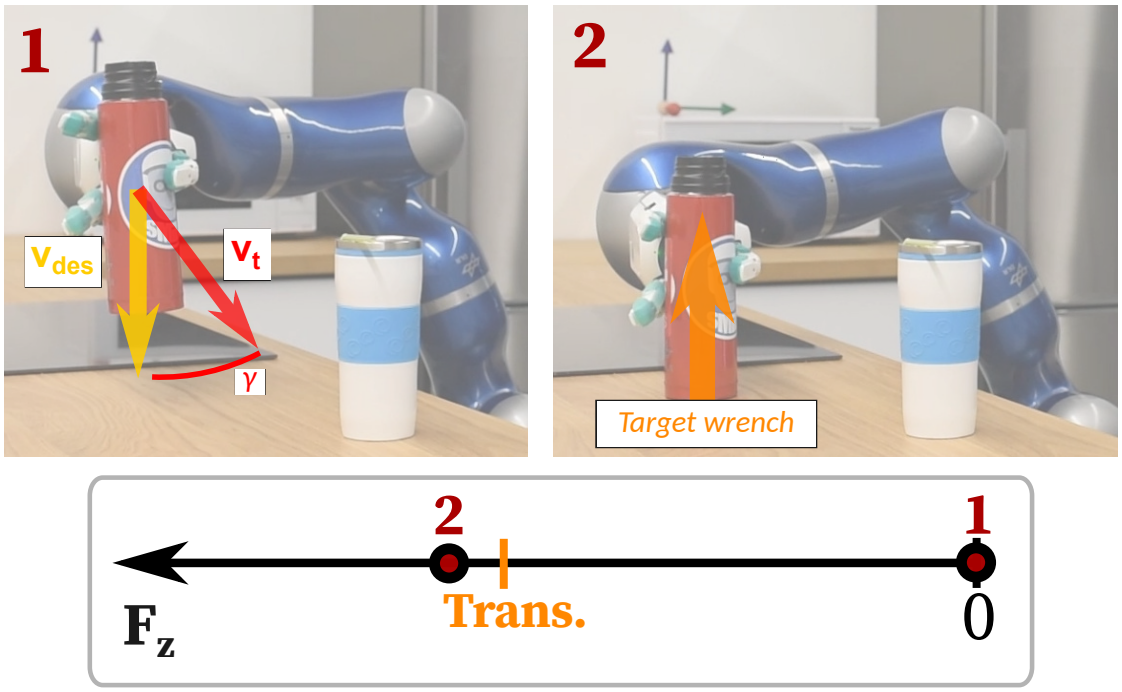



**Figure 3.4:** An example transition based on wrenches, while releasing a bottle. The target wrench for a transition is a vertical wrench shown in orange in panel 2. In order to produce this wrench, the robot should move in the opposite direction, shown in yellow in panel 1, $v_{\text{des}}$. Thus, the automaton's goal is to produce a command $x_t$ that moves the robot in a command $v_t$ (in red) that minimizes the angle $\gamma$ with respect to the the desired direction $v_{\text{des}}$. In any case, the transition only happens when the vertical force is measured over a threshold value (in panel 2). )

#### Local optimization

With a metric in Eq. (3.4) (or Eq. (3.5)), the goal is to find the best command to transition to the next state for any state and end-effector pose. However, IMs do not make any assumptions of directness or intuitiveness: for example, the command $x_t = [0\ 0\ -1]$ does not necessarily imply that the resulting pose $\mathbf{T}_t$ will be displaced vertically from $\mathbf{T}_{t-1}$. It is therefore necessary to *find* commands that minimize that loss.

This can be formalized as an optimization problem:

$$x_t^* = \arg\min_{x} \mathcal{L}_s^{s+1}(x_t). \tag{3.6}$$

Two points from Equation 3.6 are important:

1. We minimize the loss with respect to the user command $x_t$ even though the loss is directly computed from poses $\mathbf{T}_t$. Therefore, we need to use the combination of IMs and ACs from (3.1).

2. Recall that the automaton does not plan a series of commands: it greedily chooses the optimal command at each time step. In motion planning, such a greedy command generation could be prone to local minima and non-smooth movements.

There are two main reasons to explain the success of this procedure in the context of shared control templates. First, the motions are not complex, as they are generated within one state of the finite state machine. A simple point-to-point operation in input space often suffices to successfully traverse a state. Second, the search takes place in the 3D input space and not in the configuration space of the robot because it is *easier* to solve the task there. This intuition comes from (3.2): by searching on input space $\boldsymbol{x}$, using $\varphi_s$, we guarantee that the robot will stay within the task region $\mathcal{S}_s$. This greatly limits the search space, because poses in $\mathcal{S}_s$ have a higher chance of solving the task than any random pose in SE(3).

The main idea behind Shared Control is that it assists the human by allowing simple low-dimensional inputs to achieve a complex task. In our design of SCIA, the autonomous task execution benefits from this dimensionality reduction as well: not only does it allow for seamless trading of autonomy, but it also keeps the trajectories smooth, prevents collisions due to the Active Constraints, and reduces the risk of local minima. In short, we argue that the feasibility of using such a simple greedy algorithm is a feature of the shared control framework, highlighting that approaches facilitating control for humans also simplify command generation for algorithms.

Unfortunately, due to the heuristic nature of the Input Mappings and the Active Constraints, there is no clear way for inverting $\varphi_s$ from (3.1), nor is it simple to obtain a gradient. It is nevertheless straightforward to evaluate the mapping at any point. We therefore define a sampling-based evolution strategy, described next.

#### Stochastic commands

The scale (i.e., the 2-norm) of the 3D command $\boldsymbol{x}$ correlates with the length of the movement on a given time $t$ (and therefore with the speed of the action). Without loss of generality, we disentangle the scale and the direction of the command by modeling it as a unit vector in $\Re^3$. The goal here is finding the *direction that minimizes the loss*, and set the scale as a hyper-parameter of the algorithm, $\zeta \in (0, 1]$. This is analogous to the process with human commands, because the gain of the input devices (e.g., the sEMG amplifier) can be tuned according to user's preference ot specific task situations.

We generate a pool of $n$ candidate commands that we use to find the best direction in (3.6). As sampling randomly from the whole unit sphere is highly inefficient and can miss the minima, we aim to reduce the prior distribution from which we sample. We model the sampling pool as a von Mises-Fisher (vmF)[6] distribution,

$$\boldsymbol{x}_t \sim vMF(\boldsymbol{\mu}_{\boldsymbol{x}}, \kappa), \tag{3.7}$$

[6]The vMF distribution is an analogous of a 2D normal distribution projected to a sphere, and it restricts the covariance contours to be only circular.

from which we sample[7] the candidate commands. We thus introduce two parameters, the mean vector $\mu_x$ and the concentration scalar $\kappa$ (cf. Fig. 3.5).

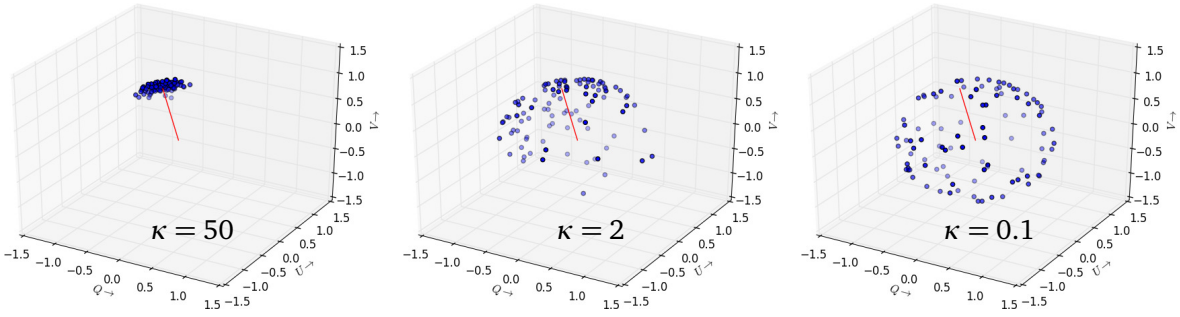



**Figure 3.5:** Effect of the concentration parameter on vMF samples ($n = 200$). The red line is $\mu_x$. **Credits:** This figure was taken from [36]. © 2021 IEEE.

### Evolution strategy

The choice of these parameters is of special interest: $\kappa$ relates to the exploration noise around the mean $\mu_x$. It is desirable to increase $n$, but it is limited by the computational power available. Fixing these parameters would either make the automaton slower or hinder the way it explores and generalizes to different scenarios.

For this reason, we devised an evolution strategy that adapts $\mu_x$ and $\kappa$ during task execution. We summarize it in Algorithm 1 and explain it next:

**Algorithm 1** Algorithm for the automaton to complete a task after an autonomy button press

**Input:** Current end effector pose $H_0$. Number of samples $n$. Percentage of elite samples $p_{elite}$. Goal state $m$, and plan through states $1...2...s...s+1...m$. Scaling parameter $\zeta$.

**Output:** Commands in user input space $\boldsymbol{x}$

1: Initialize $\mu_x$ and $\kappa$
2: **while** not on goal state $m$ **do**
3: Sample $n$ unit vectors $\boldsymbol{x}_i$ from $vMF(\mu_x, \kappa)$
4: **for** each $\boldsymbol{x}_i$ **do**
5: compute the resulting frame $H_i = \varphi_s(\zeta \boldsymbol{x}_i, H_{t-1})$
6: $\mathcal{L}i = \mathcal{D}_s^{s+1}(H_i)$
7: **end for**
8: $\boldsymbol{x}^* = \arg\min_x \mathcal{L}$
9: Take the $p_{elite}$ best performing samples as $\boldsymbol{x}_{elite}$
10: Given the current $\mu_x$, estimate a new concentration $\kappa$ from $\boldsymbol{x}_{elite}$
11: Estimate a new $\mu_x$ from $\boldsymbol{x}_{elite}$
12: Clip $\kappa$ in the range $[0.1, 100]$
13: **Issue the robot command** $\zeta \boldsymbol{x}^*$
14: **end while**

`Step zero:` Once the autonomy button is pressed, the automaton initializes $\mu_x$ as a unit

[7] We use an implementation of the spherical distribution that includes rejection sampling [59].

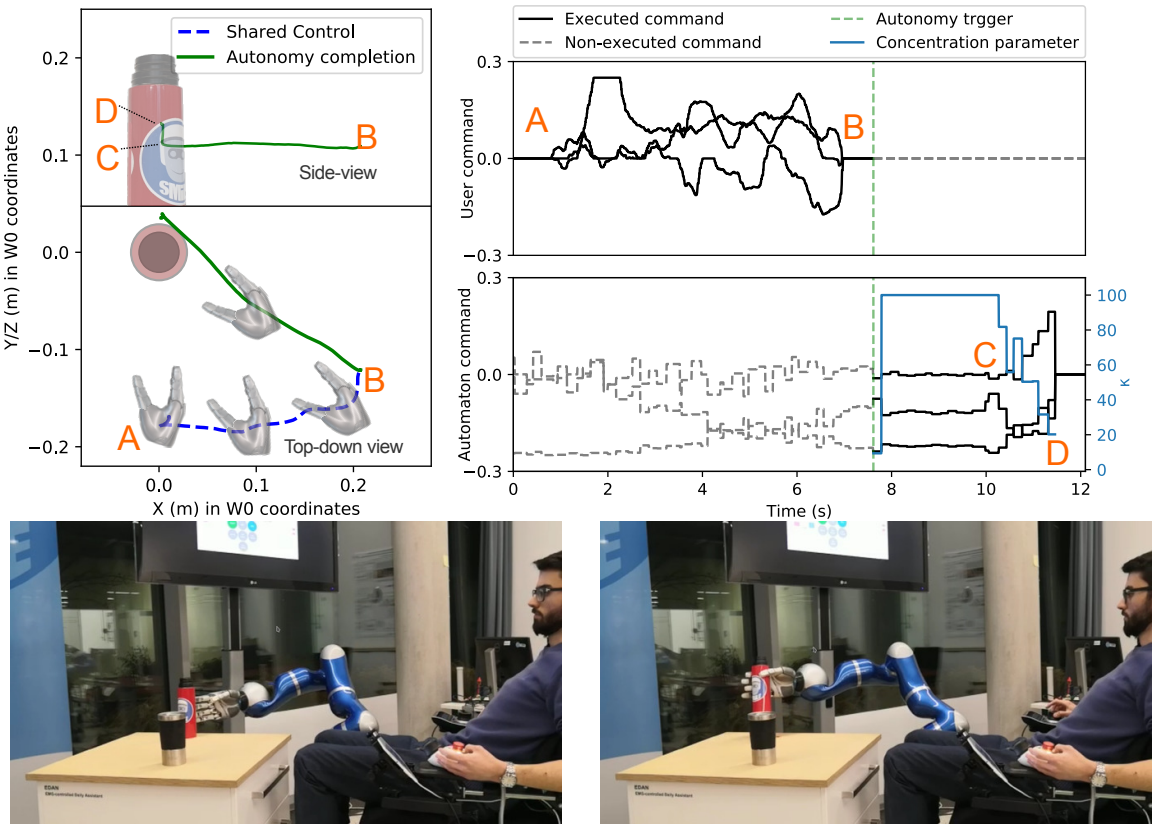



**Figure 3.6:** Grasping task on EDAN. We show the robot's measured trajectory (top left), the user/automaton commands (top right), and photos of EDAN while grasping (bottom). In segment **AB**, the user explores the Shared Control Manifold, choosing from where to grasp. Notice that the end effector rotates towards the direction of the object, as defined in the grasping SCT skill. During this time, the automaton (center) is ready in the background, and we show in gray the command it would execute if the autonomy would be started. This happens in **B**, then the automaton issues commands to finish the task. In **BC**, $\kappa$ (center) increases rapidly and stays at the maximum as the confidence in the task direction is high. In **CD**, a change of the task direction happens, and $\kappa$ briefly decreases to allow the automaton to explore more parts of the command space. The user and automaton commands are scaled by the same factor $\zeta = 0.25$. We emphasize how the user may stop the automaton commands with a click at any time during execution. **Credits:** This figure was re-scaled from [36]. © 2021 IEEE.

vector in the direction between the task frames of interest (e.g., the end-effector and the task target) and $\kappa = 3$.

`Step one:` the agent takes $n$ vMF samples from (3.7).

`Step two:` the automaton evaluates the mapping and the loss in (3.1) & (3.4) for all the samples (scaled by $\zeta$), and takes a percentage $p_{elite} = 20\%$ of the best-performing samples [8].

`Step three:` the automaton estimates a new $\kappa$ based on the spread of the elite samples given the current $\boldsymbol{\mu}_x$ (i.e., the mean that generated them), and then it estimates a new $\boldsymbol{\mu}_x$ based only on them[9]. *Intuition:* If the mean direction performs really well in minimizing the loss, these elite samples will be concentrated around the mean, the confidence of the

[8] We set the value of $p_{elite}$ the initial $\kappa$ empirically. We argue $\kappa = 3$ is reasonable as it allows to explore a significant portion of the sphere.

[9] Since there is no closed-form solution for a maximum likelihood estimator of the concentration parameter, we use an empirical approximation provided by Dhillion and Sra [52].

automaton on this command direction will be high, and $\kappa$ should increase. If it does not, the elite samples should be sparse and point towards the minima; therefore, $\kappa$ should decrease to allow the agent to explore.

`Step four:` the automaton issues the command that minimizes the loss (given (3.6), and scaled by $\zeta$), and begins again with step one by taking $n$ samples with the new parameters. The automaton will keep updating $\mu_x$ and $\kappa$ throughout the task[10]. Furthermore, we clip $\kappa$ to the range $[0.1, 100]$ to prevent excessive crunching or spreading.

To illustrate these concepts, we show the automatic completion of a grasping task on EDAN in Fig. 3.6. The evolution strategy described here rapidly converges into a command direction that solves the task, but it still allows the robot to react when the task requires a change of direction, and thus explore (and generalize to) new scenarios.

### 3.2.2 Task completion

Given that the automaton can now traverse a single SCT state, the process for automatic task completion is as follows: During Shared Control, the robot has symbolic knowledge of the user goal. In practice, as discussed on Section 2.3.5, SCTs from the world model are typically activated based on either a UI or based on objects' affordances. Given this goal and information, the robot can follow the SCT and knows the set of transitions necessary to achieve the desired goal. As an example, if during the pouring task in Fig. 3.1 the goal is to *rotate the tip* (state 3) and the robot is in *translational control* (state 1), the robot will get a plan: first go to state 2, then go to state 3.

If the human is commanding the actions of the robot in Shared Control, the automaton will be ready to start finishing the task. If the human presses the SCIA button, the robot immediately queries this plan. Instantly, it retrieves the current state $s$ and the sets of transitions it needs to reach the goal; then, it starts producing input commands autonomously as explained before. Since the robot does not require re-planning, we posit that this transition is seamless. We show this in Fig. 3.6. The robot starts moving in the direction that solves the task as soon as the user presses the autonomy trigger [11]. Therefore, the SCIA procedure circumvents what is a known problem in adaptive autonomy, precisely that autonomous agents cannot keep track of what the human is doing during teleoperation [105]. Additionally, users can seamlessly regain control from the robot. This will happen if they request it within an SCT execution (e.g., with another button press) or if the automaton finishes the task.

[10] Only to be reset in specific cases like task completion, or moments in which the user input is blocked to wait for the robot to finish another motion.

[11] I mention the pitfalls of the local planner in Section 3.4.

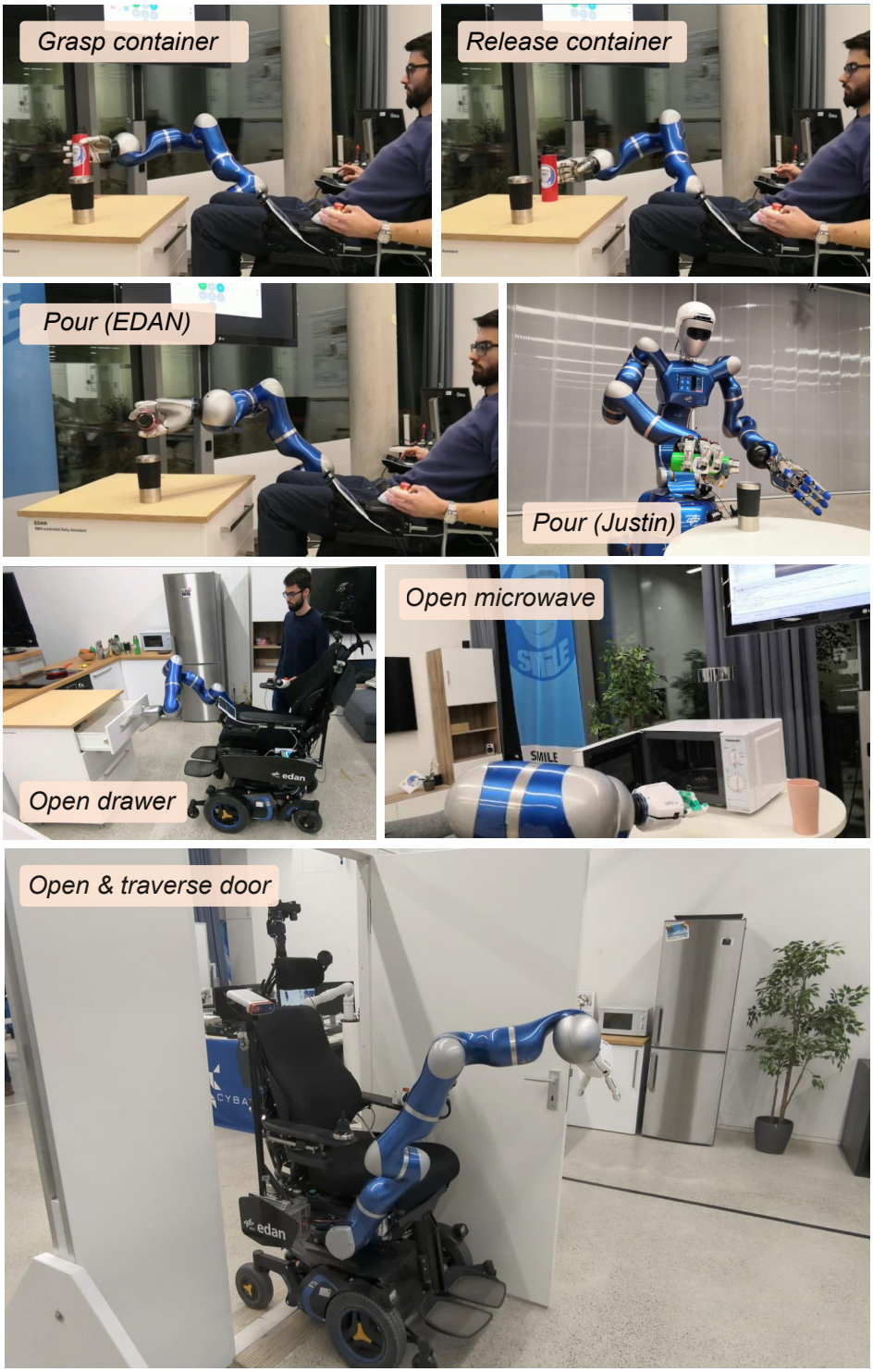



**Figure 3.7:** Daily living tasks our robots can complete using the automaton. Shown are EDAN and Rollin' Justin performing a set of daily living tasks. Different versions of EDAN have been used, for instance using different hands. The SCTs used represent a combination of dynamic (e.g. the microwave button) and kinematic (e.g. pouring frames) transitions. I highlight that the automaton does not "learn" these tasks, it generates commands zero-shot based on a pre-existing SCT. **Credits:** Both pouring images as well as the container grasp, container release and open drawer were taken from [36] (© 2021 IEEE). The microwave photo was taken from [37] (© 2022 IEEE)

.

### 3.2.3 Example tasks and robots

The simple procedure described in the previous subsections can be used to enable robots to autonomously complete different tasks. Currently, our robots can use Shared Control with Integrated Autonomy on several activities of daily living. Fig. 3.7 shows seven tasks deployed with SCIA in our robots EDAN and the humanoid robot Rollin' Justin [28].

#### Whole-body control:

SCTs support whole-body control (WBC) of the robot platform while the user is in control, which increases the workspace and reachability of the robot [88]. Initially, the SCT outputs a goal frame, which is used as the target for the EE. If the user command leads the robot through certain pre-established boundaries (e.g., if the robot is reaching too much to the front and is running out of space), spring forces act on the robot platform to move it forward or backward, effectively coordinating the EE and the platform motion, and providing the EE more reachability [88].

As WBC is provided on top of pre-existing SCTs, it is available to SCIA "off-the-shelf". Thus, the automaton is able to guide the robot in complex tasks that require the orchestration of the EE and the platform, such as opening a drawer (while moving the wheelchair back to provide more space), and opening & traversing a door. Both tasks are shown in Fig. 3.7.

#### Anecdotal evidence: deployment as a supervised autonomy demonstrator

On July 20-21 2023, EDAN participated in a workshop for the project SMiLE2getherGaPa, where a prototype for the robot operating in supervised autonomy was shown at an elderly care facility in Garmisch-Partenkirchen, Germany.

During the workshop, our team showed diverse grasping tasks, including an autonomous version of the `clean-up shelf` task of the 2023 Cybathlon challenge [53] [12], where each of the tasks was completed by the automaton. The task consisted of picking a set of items (a cube, a toothbrush-like tube, a dice, and a disk) from a 2x2 shelf as in Fig. 3.8 (A), and placing them on top of the shelf, as depicted in Fig. 3.8 (C). The position of the items was fixed within their cubicle, but they could be positioned in any cubicle. For our demonstrator, we used pre-existing SCTs from the Cybathlon challenges. To activate the tasks, we used a tablet, shown in Fig. 3.8 (B), which automatically started the automaton.

On Fig. 3.9, I illustrate a full execution of the `clean-up shelf` task by the automaton, i.e., picking and placing all items from the state in Fig. 3.8 (A) to the state in Fig. 3.8 (C). For each of the tasks, one team member clicked one of the actions of the tablet, starting the automaton to pick and place the object. The entire sequence takes about 3 minutes and 20 seconds and involves four separate SCTs, one for picking each item. These SCTs enabled

[12] The EDAN team won this challenge, in shared control [53].

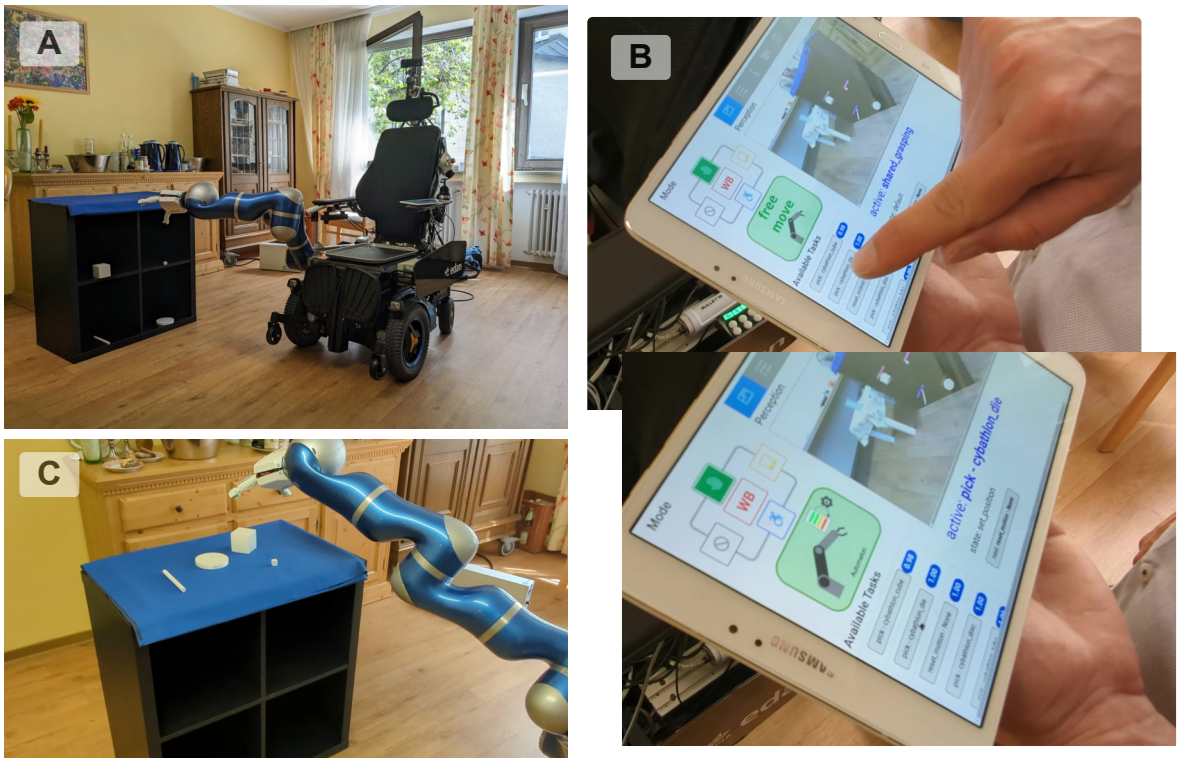



**Figure 3.8:** Detail of the `clean-up shelf` experiment demonstrated in an elderly care facility. The task consists of grasping the items from the shelf (shown in **A**) and placing them on top (shown in **C**). The task and the automaton were triggered via a tablet interface, shown in **B.**, where we depict a prototype user interface that shows the user the automaton is active (lower right, in green). Thanks to J. Vogel and G. Quere for the assistance with this demonstrator.

complex manipulation of the items: for instance, the disk SCT guided the EE to slide the object out of the shelf before grasping it. Furthermore, some of the transitions were based in contact forces with the shelf.

## 3.3 Experiments

In [36] my colleagues and I presented and discussed two sets of experiments with our robots EDAN and Rollin' Justin (see a description of the robots in Section 2.4). These experiments aim to show how SCIA enables Supervised Autonomy after a button press, focusing on testing efficiency (set 1) and robustness (set 2) of the autonomous agent. Furthermore, they showed the generality of the method by showing an example of two robots using the same skill.

### 3.3.1 Description:

Four tasks from Fig. 3.7 were used, described in detail on each experiment. All runs of the automaton start in Shared Control and switch to Supervised Autonomy after a button press

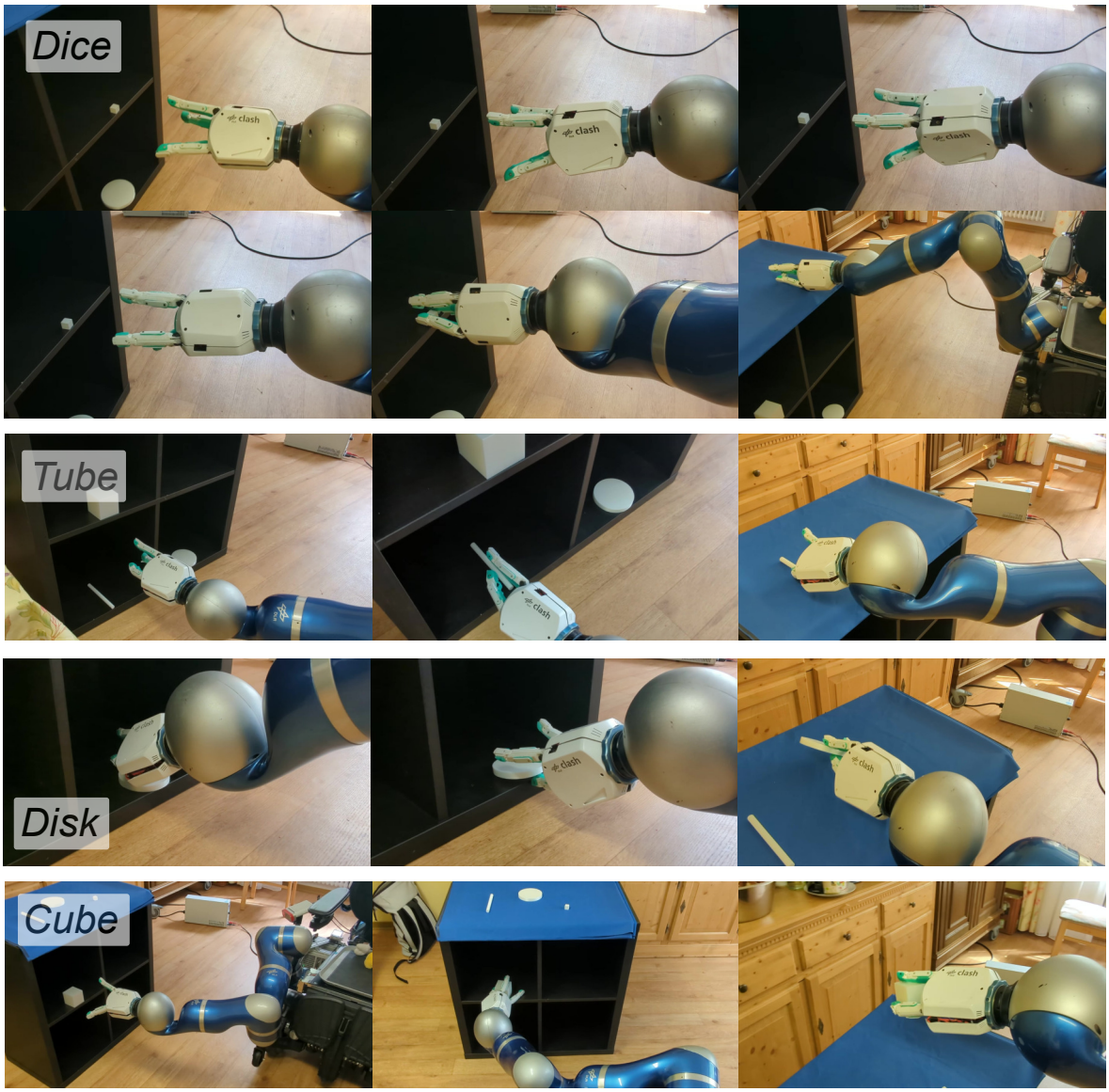



**Figure 3.9:** Entire sequence for the task `clean-up shelf` with EDAN. Each individual subtask (in order, dice, tube, disk, and cube) was started with the tablet, and completed by the automaton in supervised autonomy. I highlight some of the complex manipulation steps the automaton achieves thanks to the SCT, like a reorientation of the EE while grasping the dice, or sliding the disk before grasping it.

at the start of the trial. The experiments on EDAN were conducted with an RGB-D-based perception pipeline for detecting and estimating the pose [196] of the bottle, the mug, and the drawer.

### 3.3.2 Experiment set 1: speed test

A factor that could impact acceptance of SCIA is the speed of the autonomous task execution, as we hypothesize that users could dislike the autonomous execution if it takes much longer to complete than in Shared Control mode. As a first step to analyze this problem, an experi-

ment was conducted on EDAN to estimate the difference in the speed execution between the automaton and an expert user.

To measure the difference, 20 successful trials were collected: 10 successful trials with the automaton (triggered by the experimenter), and 10 successful trials with an expert human (EH). EH was part of the authors list of the original study [36]. At the time of the study, they did not have any motor impairment and had previous experience with SCTs. Furthermore, EH was instructed to solve the task as quickly as possible.

**Task description:** The task was a grasp & pour experiment on EDAN (Fig. 3.7). A trial consisted on grasping the bottle from the table, lifting it, pouring actual water in the mug, and retreating from the mug. *Initialization*: Hand, mug, and bottle in fixed positions. *Success criteria*: No collision with the bottle, no water spilling, and water is transferred from the thermos to the mug.

**Results:** The results are shown in Table 3.1. Task times are calculated so that a fair comparison is possible, for instance, by leaving out those SCT skill states in which the experimenter had to press the autonomy button, and also those in which EDAN would move without a user command (for example, states that silence the user input while the robot would orient itself).

**Table 3.1:** Comparison of execution times (s), $\mu \pm \sigma$ over 10 trials. **Credits:** This table was taken from [36]. © 2021 IEEE.

| Task | Expert human | Automaton |
|---|---|---|
| `Grasp` ($p > 0.05$) | $1.87 \pm 0.25$ | $2.06 \pm 0.12$ |
| `Pour`, approach ($p < 0.001$) | $7.10 \pm 1.08$ | $5.71 \pm 0.08$ |
| `Pour`, retreat ($p > 0.05$) | $8.22 \pm 1.26$ | $8.70 \pm 1.01$ |

**Discussion:** In one case (Pour, approach), the automaton was significantly faster than the EH (non-parametric unpaired two-sided Mann–Whitney U-test), and in the others there was no significant difference. This confirms that the Automaton *performs at least as good* as the human or better, which (we expect) to be an important factor for acceptance. However, we did not test this hypothesis.

### 3.3.3 Experiment set 2: activities of the daily living

Another key aspect of seamless autonomy is the ability to complete a task successfully, from a good range of starting positions where the user could trigger it. We aim, therefore, to evaluate the automaton's performance on different tasks while being triggered from controlled bounding boxes, both on EDAN and Rollin' Justin. Since the SCTs automatically orient the robot end-effector during the task, we fixed the starting orientation of the robots.

**Task description:** Next is a description of the task steps, success criteria, and initialization.

`EDAN pour`. *Task steps*: While a bottle is grasped, approach the mug, pour a small rubber ball from the bottle to the mug, and then move away. *Initialization*: Mug in random position on a bounded surface in the table. EDAN's hand starts in a random position on a bounded surface. Wheelchair position fixed. *Success Criteria*: The rubber ball is transferred successfully, no significant collision between the robot (or the bottle) and the mug.

`Justin pour`. *Task steps*: Same as EDAN, using the same SCT skill with little parameter adaptation. `Initialization`: Mug and platform position fixed. Justin hand starting in a random position in a bounded surface. *Success Criteria*: Same as EDAN.

`EDAN pick & place`. *Task steps*: Grasp the bottle from the table, lift it a few centimeters, move down, release it, and move away. *Initialization*: Bottle in a random position on a control surface in the table. EDAN's hand starts in a random position on a bounded surface. Wheelchair position fixed. *Success Criteria*: No collision with the bottle and bottle grasp not extremely tilted, in the experimenter's judgment. The bottle is upright after releasing.

`EDAN drawer opening`. *Task steps*: Open the drawer a few centimeters with active whole-body control. *Initialization*: Wheelchair position arbitrarily set by the experimenter. Hand position fixed with respect to the wheelchair (not to the drawer). *Success Criteria*: Open drawer, robot forces not exceeding a safety threshold.

**Results:** Table 3.2 reports the success rate of both robots. Similarly, the robot trajectories for EDAN's pick & place and pouring tasks are shown in Fig. 3.10, which illustrates the bounded surfaces we used.

**Table 3.2:** Success rate of Activities of the Daily Living. The task was performed on EDAN unless stated otherwise. **Credits:** This table was taken from [36]. © 2021 IEEE.

| Task | Number of trials | Successful trials |
|---|---|---|
| `Pour` (with EDAN) | 45 | 44 (98%) |
| `Pour` (with Justin) | 15 | 14 (93%) |
| `Grasp & Release` | 15 | 14 (93%) |
| `Drawer` opening | 15 | 15 (100%) |

*Discussion:* Table 3.1 and Fig. 3.10 highlight that the automaton is effective in performing tasks of daily living, and we validate our approach with two real robotic systems. This illustrates the capabilities of the system to allow switching from Shared Control to Supervised Autonomy and finish tasks in different settings, including a moving platform with whole-body control, and demonstrates the generality of the concept by transferring it to a different robotic system. This switch is seamless. In comparison, if the system were to stop the SCT and switch to an autonomy representation, switching would be slow since a full motion plan is required, and back-and-forth transitions are difficult as the semantic state could not be easily shared between representations.

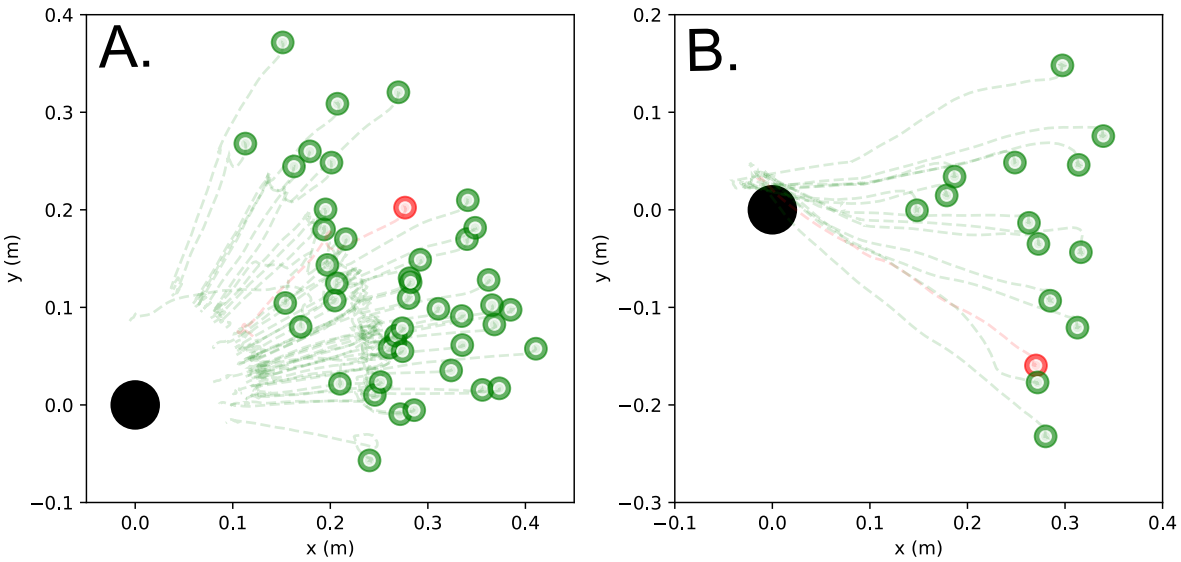



**Figure 3.10:** Robot motions (top view) during two activities of the daily living. Depicted are the starting EE positions in circles (green in successful trials, red otherwise). **A.** pouring motion relative to the detected mug position (black circle, centered on (0,0)). **B.** grasping motion relative to the bottle position (idem). Note that the target (mug and bottle, respectively) was not located in a fixed position but in a random place of the workspace (see text for details), and we centered it for illustration purposes. Not shown is the retreat from pouring motion in **A.** and the release motion in **B. Credits:** This figure was taken from [36]. © 2021 IEEE.

However, we have found empirically that the automaton's success depends on whether it is started by the user in a feasible position or elsewhere. As an example, if the pouring movement in Fig. 3.10A. would start south-west of the control volume, it would often lead to task failure. This is despite the fact that the automaton could find a task-space trajectory towards the goal, and that it would lie on the task region, because the robot would have had limited manipulability, or the motion results in self-collision. We discuss these shortcomings in Section 3.4.

## 3.4 Discussion and outlook

This chapter proposed a novel concept for human-triggered adjustable autonomy in the context of assistive robotics, by expanding the Shared Control Templates framework to include Shared Control with Integrated Autonomy. Using the same action representation, users can seamlessly switch between Shared Control and Supervised Autonomy for task completion. Critically, such a framework allows the autonomous agent to keep track of the actions of the human (in Shared Control) and switch instantly if needed.

However, as noted in the experiments, two problems exist with the original formulation of SCIA:

- SCTs reason in task space and are object-centric. This facilitates the transfer of skills between robots. However, it entails that they are unaware of kinematic limitations of spe-

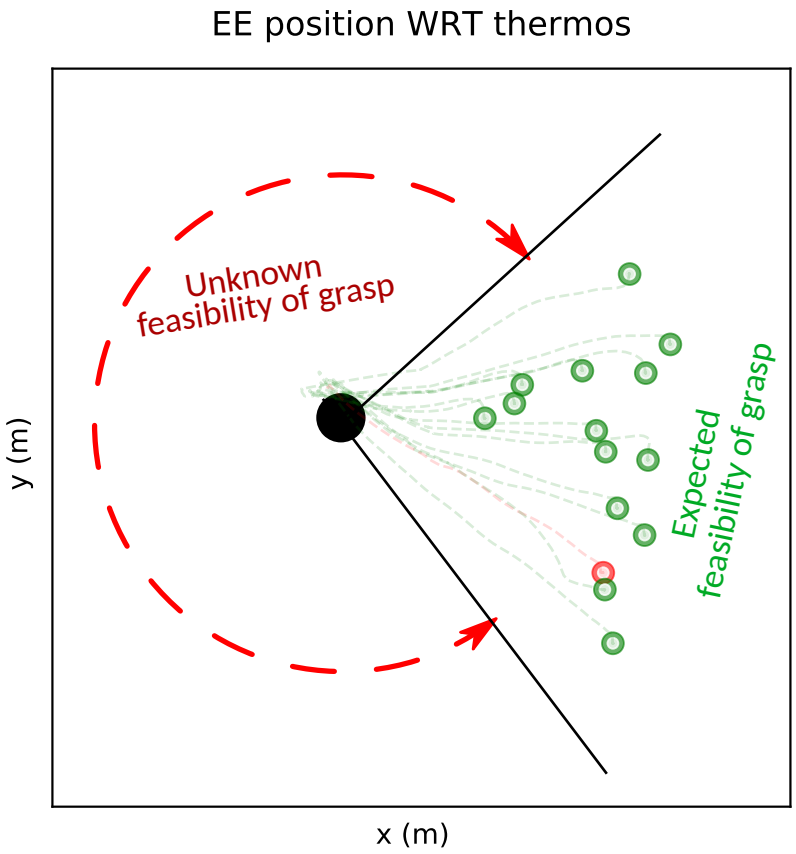



**Figure 3.11:** Why are feasibility checks important in the context of SCIA? Here, I show again the grasping data from Fig. 3.10(B) but highlight the bounded region from which the task was started. In the other region, the feasibility of the task was not guaranteed, as kinematic limitations could arise, impairing any task in both shared control and supervised autonomy. The conclusion is that there is a need to *assess the feasibility of the task* before it is started, which is the focus of the next chapter.

cific robots, for instance, limited manipulability, workspace, joint limits, and obstacles, which are included in conventional motion planners like CHOMP [211] or STOMP [94]. This causes task success to be dependent on the user activating the automaton in a feasible position, as shown in Fig. 3.10 and the subsequent discussion. However, in practice, the user may want to start a SCIA task from *anywhere* in the workspace. It would be worthwhile to remove from them the burden of assessing the feasibility of a given task. For instance, it would be ideal to start the automaton anywhere around the target object, as depicted in Fig. 3.11. Studying this problem is the focus of Chapter 4, where a solution is proposed.

- Automaton triggers operate on single tasks, and thus tasks with longer event horizons are not possible. As an example, it would be a more intuitive interface if the user would provide the robot with the multi-SCT goal such as "`serve a drink`", instead of individually activating the grasping, the pouring and the releasing tasks in Fig. 3.7. Likewise, each one of the item subtasks for the `clean-up shelf` demonstrator in Fig. 3.9 had to be started independently. Solving this problem is the focus of Chapter 5.

# 4

# Feasibility checks before shared control tasks

*Despite being supervised by a human, robotic actions in shared control can still go wrong. The challenge is to give the best possible support to the human, preventing failure situations. This means to simultaneously provide support for completing daily living tasks, and to enable sufficient freedom of movement to foster user empowerment, but at the same time preventing the robot from running into kinematic limitations such as obstacles, unreachable areas, or loss of manipulability due to joint limits.*

*This chapter examines a framework for performing feasibility checks before executing a shared-control task. We activate shared control only if a task is deemed feasible, and refine the task regions by excluding paths that are infeasible, e.g., due to obstacles or kinematic limitations. This reduces task failures, whilst still ensuring freedom of movement. We evaluate our framework on a set of daily living tasks with EDAN.*

***This chapter originates from a paper for the IEEE Robotics and Automation Letters, and I include some literal text portions and figures. However, it has been reworked to fit as a chapter within the thesis concept, and also features a few extensions such as Section 4.2.3 and some figures. See:[38]. Copyright note in the footnote*** [1]**.**

From the robot's side, multiple things can go wrong during a shared control task: it may lose manipulability due to a joint limit, fall into a singularity, or hit an obstacle. The goal of this chapter, summarized in Fig. 4.1, is to provide a framework that checks the feasibility of conducting a task in shared control *before* it is executed, using a kinematic simulation of the task. The framework is based on the SCTs and a constraint representation derived from the Task Space Regions (TSRs [21], described in detail in Section 4.1). If the task is deemed

[1]Whenever parts of the manuscript are taken literally: © 2024 IEEE. Reprinted, with permission from the senior author, from S. Bustamante, I. Rodríguez, G. Quere, P. Lehner, M. Iskandar, D. Leidner, A. Dömel, A. Albu-Schäffer, J. Vogel, F. Stulp, Feasibility Checking and Constraint Refinement for Shared Control in Assistive Robotics, IEEE Robotics and Automation Letters, Sept. 2024.

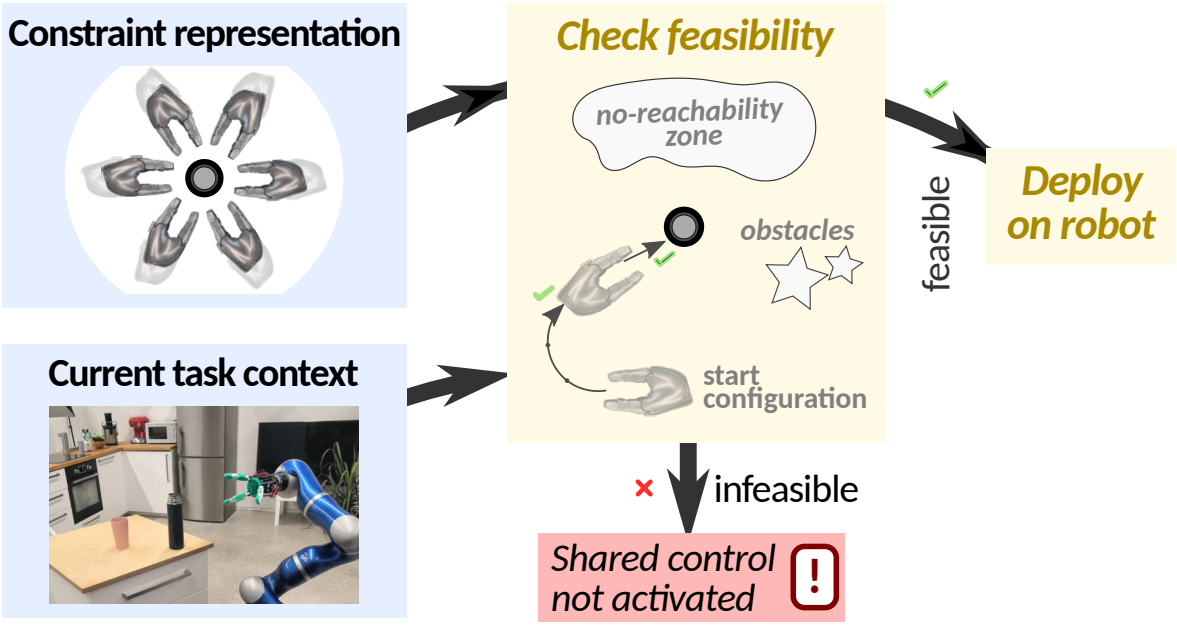



**Figure 4.1:** Summary of the feasibility checking framework. Shared control is only activated after its feasibility with respect to obstacles and kinematic constraints has been confirmed. If it is feasible, task constraints are refined (not shown), and shared control is instantiated with the refined constraints. **Credits:** This figure was taken from [38]. © 2024 IEEE.

infeasible, it stops its activation, preventing the user from starting tasks that are predicted to fail eventually. But if it is feasible, it further *refines* the shared-control constraints of the SCTs, supporting the user to avoid kinematic limitations *during* the assisted task. The resulting SCT is deployed on the robot to support the user.

This chapter starts in Section 4.1 with an overview of the method. Particularly, I describe the TSRs, and how they can be used in shared control via a new ancillary representation to the SCTs, the *refined* TSRs (rTSRs). Then, in Section 4.2 I present the method in detail, including some qualitative examples on EDAN. After that, in Section 4.3 I present a series of experiments on EDAN and in simulation, answering a series of research questions. Finally, in Section 4.4 I draw a discussion of the limitations, its possible extensions, and an outlook to the next chapters.

## 4.1 Preliminary: Task Space Regions for shared control

This section provides an overview of the representations used to support the method. The ultimate aim is to refine the constraints in an SCT so that it prevents users from colliding with obstacles or encountering kinematic issues. We further avoid activating SCTs altogether if no collision-free paths exist.

### 4.1.1 Background: Task Space Regions

TSRs are a well-established constraint representation for motion planning [21]. A TSR $\mathcal{S}$ describes an end-effector constraint (subset of SE(3)) given by the allowed motion of a coordinate frame $w'$ in the TSR origin $w$. For example, as a cylindrical container is symmetric around the axis, its grasp frame can be located anywhere around it, as depicted in Fig. 4.2.




**Figure 4.2:** A task space region for grasping a cylindrical container. Since the container is symmetric, it can be grasped from any direction.

Formally, a TSR consists of three elements,

$$\mathcal{S} = \{ \mathbf{T}_w^0,\ \mathbf{B}^w\ , \mathbf{T}_e^{w'} \}, \tag{4.1}$$

where $\mathbf{T}_w^0$ is the fixed transform from the world origin 0 to $w$, $\mathbf{T}_e^{w'}$ is a fixed frame (called an *end-effector offset*) in the coordinates of $w'$, and $\mathbf{B}^w$ is a $6 \times 2$ matrix of bounds for the allowed movement of $w'$ in the coordinates of $w$, using a Roll-Pitch-Yaw Euler angle convention:

$$\mathbf{B}^w = \begin{bmatrix} x_{\min} & y_{\min} & z_{\min} & \psi_{\min} & \theta_{\min} & \phi_{\min} \\ x_{\max} & y_{\max} & z_{\max} & \psi_{\max} & \theta_{\max} & \phi_{\max} \end{bmatrix}^\intercal \tag{4.2}$$

$\mathbf{B}^w$ defines a region of constrained poses $\in$ SE(3). We can take samples from the region by obtaining values from the ranges of $\mathbf{B}^w$. A sample is a vector

$$\boldsymbol{b}_i' = \begin{bmatrix} x_\mathrm{i} & y_\mathrm{i} & z_\mathrm{i} & \psi_\mathrm{i} & \theta_\mathrm{i} & \phi_\mathrm{i} \end{bmatrix}^\intercal \tag{4.3}$$

which can be converted to a transform $\mathbf{T}_{w_i'}^w = f(\boldsymbol{b}_i')$, as described in [21]. Then the corresponding EE pose sample with respect to the world origin $\mathbf{T}_{e_i}^0$ is obtained with

$$\mathbf{T}_{e_i}^0 = \mathbf{T}_w^0\ \mathbf{T}_{w_i'}^w\ \mathbf{T}_e^{w'} \text{ with } \mathbf{T}_{w_i'}^w = f(\boldsymbol{b}_i') \quad \text{compare Eq. (4.1)} \tag{4.4}$$

where $\mathbf{T}_w^0$ and $\mathbf{T}_e^{w'}$ are independent of the samples taken from $\mathbf{B}^w$.

**A comment on the notation:** In Eq. (4.4), the resulting EE pose is $\mathbf{T}_{e_i}^0$, not $\mathbf{T}_e^0$. This is because the pose depends on the sample $\boldsymbol{b}_i'$ of the TSRs. Thus, while there is only one offset $\mathbf{T}_e^{w'}$, there are infinitely many poses $\mathbf{T}_{e_i}^0$ that solve the constraints of the TSR.

**Running example:** Fig. 4.3 illustrates the relevant coordinate frames of a TSR in the context of a bottle grasping task, and two samples taken from the TSR with Eq. (4.4). The TSR represents that the container can be approached from any direction, as it is symmetric.

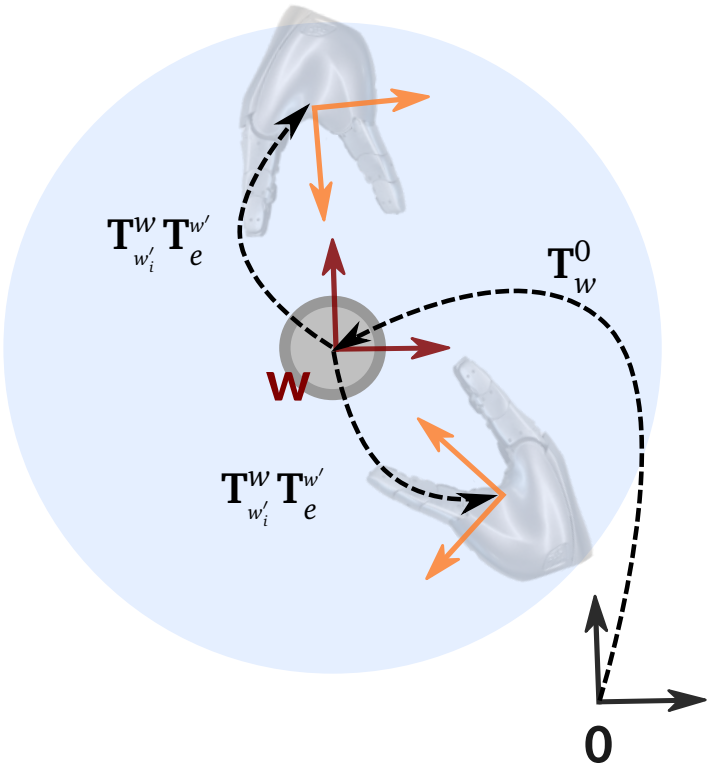



**Figure 4.3:** Illustration of two samples of the TSR for grasping a cylindrical container (top view). Exploiting the vertical symmetry, the frame $w'$ (not shown) can rotate around the vertical axis of $w$. While the transformation $\mathbf{T}_e^{w'}$ is fixed, two samples of $\mathbf{T}_{w_i'}^{w}\mathbf{T}_e^{w'}$ will yield different transformations. **Credits:** This figure was modified from [38]. © 2024 IEEE.

### 4.1.2 Representing Active Constraints in SCTs with TSRs

The reader may recall from Section 2.3 that SCTs represent active constraints *implicitly* as projection functions, given by Eq. (2.6). Therefore, they do not provide a suitable (explicit) search space within which to conduct feasibility checks. It is therefore necessary to use an ancillary representation to provide this search space. As one of the main motivations behind TSRs is precisely to provide such an explicit search space in a motion planning context [21], TSRs are a suitable formalism for representing these constraints explicitly.

Consider again the running example in Fig. 4.3. While approaching a cylindrical container, $\mathbf{T}_w^0$ centers the frame of reference on the container, and $\mathbf{T}_e^w$ projects from the bottle to the end-effector[2]. $\mathbf{B}^w$ would then represent that $(a)$ the container can be approached from any yaw angle, i.e. $\phi \in [-2\pi, 2\pi]$; $(b)$ the relative roll and pitch angle of the end-effector to the bottle is fixed, i.e. $\psi = \theta = 0$; and $(c)$ the appropriate height of the end-effector depends on the height of the bottle $z \in [z_{\min}, z_{\max}]$. $\mathbf{B}^w$ would thus be

$$\mathbf{B}^w = \begin{bmatrix} 0 & 0 & z_{\min} & 0 & 0 & -2\pi \\ 0 & 0 & z_{\max} & 0 & 0 & 2\pi \end{bmatrix}^\intercal . \tag{4.5}$$

As another example, when grasping a drawer with a long horizontal handle, the end-effector can be placed anywhere along the handle, but the $x$ and $z$ coordinates are fixed, as well as the orientation:

$$\mathbf{B}^w = \begin{bmatrix} 0 & y_{\min} & 0 & 0 & 0 & 0 \\ 0 & y_{\max} & 0 & 0 & 0 & 0 \end{bmatrix}^\intercal . \tag{4.6}$$

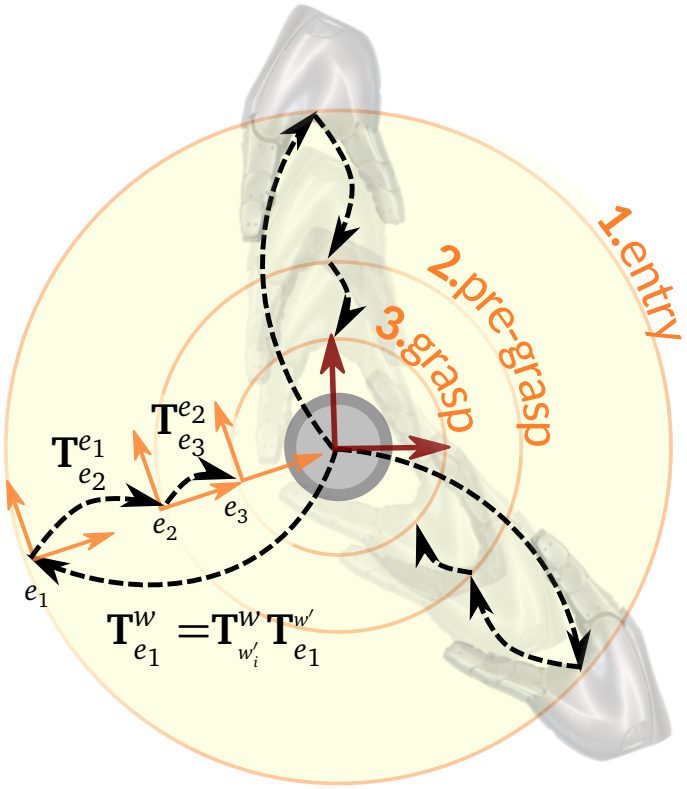



**Figure 4.4:** SCTs may have multiple subsequent states, for instance, from the entry to the pre-grasp, and from the pre-grasp to the grasp. This is represented by a chain of transformations in the TSR. A post-grasp pose ($\mathbf{T}^{e_3}_{e_4}$) lifting the object was used in the experiments, but is not illustrated. **Credits:** This figure was modified from [38]. © 2024 IEEE.

[2]These example tasks are illustrated in Section 4.3.

SCTs comprise multiple states with different active constraints. For the running example, for instance, a pre-grasp and a grasp are also represented and traversed in sequence in the different states, as shown in Fig. 4.4. To represent this with a TSR, the standard TSR representation in Eq. (4.1) can be extended by adding $n \geq 0$ transformations to the TSR as subsequent EE offsets, i.e.

$$\mathcal{S}^* = \{\ \mathbf{T}^0_w,\ \mathbf{B}^w\ , \{\mathbf{T}^{w'}_{e^1}, \mathbf{T}^{e^1}_{e^2}, \ldots, \mathbf{T}^{e^{n-1}}_{e^n}\}\}. \tag{4.7}$$

Given a sample $\boldsymbol{b}'_i$ from $\mathbf{B}^w$ which is converted to $\mathbf{T}^w_{w'_i}$, the first pose in the chain is computed with Eq. (4.4), i.e. $\mathbf{T}^0_{e^1_i} = \mathbf{T}^0_w \mathbf{T}^w_{w'_i} \mathbf{T}^{w'}_{e^1}$, and all subsequent poses in the chain are computed with:

$$\mathbf{T}^0_{e^j_i} = \mathbf{T}^0_{e^1_i} \prod_{k=2}^{j} \mathbf{T}^{e^{k-1}}_{e^k}. \tag{4.8}$$

In the above, $\mathbf{T}^0_{e^j_i}$ refers to the transformation from the origin to the $j^{\text{th}}$ via-point and $i^{\text{th}}$ random sample from $\mathbf{B}^w$, as illustrated in Fig. 4.4 for 3 via-points and 2 samples.

### 4.1.3 Refined TSRs

TSRs are object-centric. Thus, if the object is moved, the coordinate frame $w$ moves as well, and thus all derived frames. However, due to being only object-centric, TSRs as described so far do not take kinematic limitations into account. Thus, not all sampled end-effector poses $\mathbf{T}^0_{e^j_i}$ (for a sample $\boldsymbol{b}'_i$) may be reachable in practice, for instance due to obstacles or joint limits. Applying the constraints from such a TSR would thus not necessarily preclude the user from violating kinematic limitations, which is to be avoided.

We therefore *refine* the matrix $\mathbf{B}^w_{\text{ref}} \subseteq \mathbf{B}^w$, such that samples taken from $\boldsymbol{b}'_i \sim \mathbf{B}^w_{\text{ref}}$ result in paths (through all via-points $\mathbf{T}^0_{e^{j=1\ldots n}_i}$) that do not violate kinematic limitations given the current task context.

$$\mathcal{S}^*_{\text{ref}} = \{\ \mathbf{T}^0_w,\ \mathbf{B}^w_{\text{ref}} \subseteq \mathbf{B}^w,\ \{\mathbf{T}^{w'}_{e^1}, \mathbf{T}^{e^1}_{e^2}, \ldots, \mathbf{T}^{e^{n-1}}_{e^n}\}\}. \tag{4.9}$$

The outcome of this procedure is illustrated in Fig. 4.5. The procedure for doing so is explained in detail in Section 4.2.

### 4.1.4 Instantiating an SCT from a refined TSR

An SCT is a template in which the active constraints are instantiated from a TSR. The procedure explained in this section applies to unrefined and refined TSRs alike, and is used to instantiate refined active constraints in Section 4.2.2.

The bounds $\mathbf{B}^w$ (or $\mathbf{B}^w_{\text{ref}}$) of the TSR define a region in end-effector space. This region is converted to a projection function that transforms poses *outside* of the region to poses *on* the

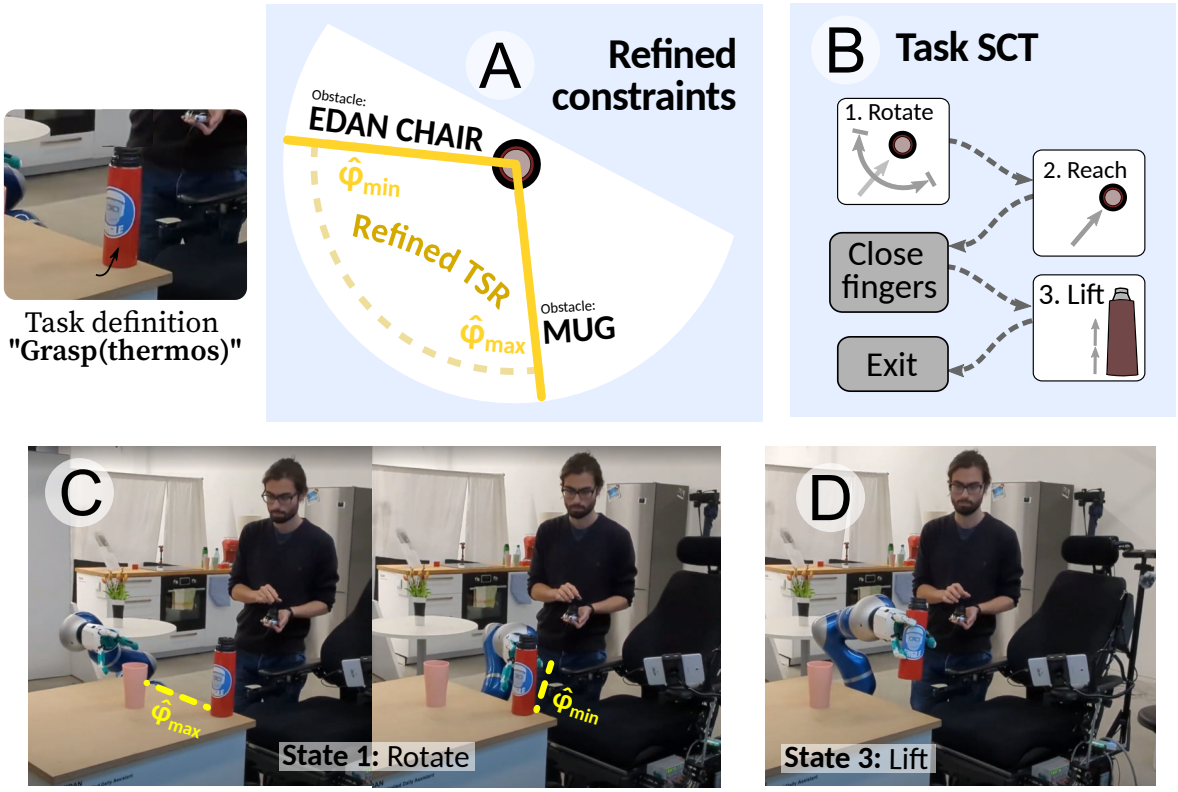



**Figure 4.5:** Refined constraints in the context of the SCT for grasping a thermos. (**A**) While *in principle* the object can be grasped from anywhere around its circumference, *in practice* some parts of the workspace can cause kinematic limitations, such as obstacles. In the example, a mug and the EDAN chair itself are obstacles in the grasping task. (**B**) The task is defined given an SCT that includes states for rotating around the thermos, reaching for it, closing the fingers of the arm, and lifting it. (**C**) In the *Rotate* state, the constraints of the refined constraints are applied by the SCT, preventing the user from hitting obstacles. (**D**) The user finishes the task in state **Lift**. © 2025 The thesis author.

region boundary, a common approach for active constraints [29]. This is straightforward, as the boundaries of the region are linear, e.g., $z > z_{\text{min}}$, see Fig. 4.6 for an illustration, and Fig. 4.5 for a real-world example of the outcome of this procedure.

Note that the first state is also responsible for guiding the end-effector from the activation pose to $\mathbf{T}_{e^1}^{w'}$, see the dashed line. From $\mathbf{T}_{e^1}^{w'}$ onward, the user moves freely within the AC. The subsequent frames of reference $\{\mathbf{T}_{e^2}^{e^1}, \mathbf{T}_{e^3}^{e^2}, \dots\}$ define ACs in the subsequent SCT states.

## 4.2 Feasibility checking and constraint refinement for shared control

This section describes the algorithm for acquiring $\mathbf{B}_{\text{ref}}^{w}$ from $\mathbf{B}^{w}$, which results in a refined TSR given by Eq. (4.9).

This algorithm is run *before* activating shared control, and consists of two main parts (illustrated in Fig. 4.7): A) ***Feasibility checking:*** Finding one path that does not violate kinematic limitations. If there is no such path, shared control is not activated. B) ***Constraint***

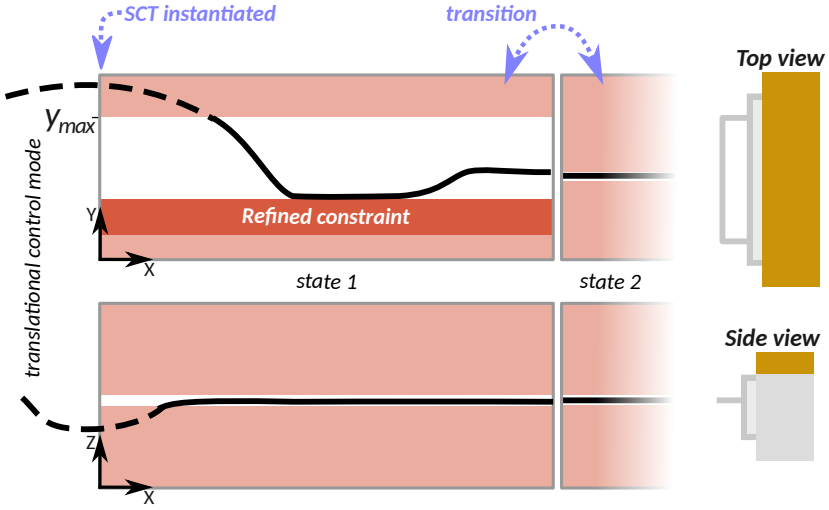



**Figure 4.6:** Schematic of an instantiated SCT for opening a drawer. The active constraints bring the robot from the activation pose to $y_{\max}$, and enable the user to move freely in the $xy$ plane (top view), but fix the EE height (side view). $xy$ not drawn to scale. A refined constraint (dark red) added from the refined TSR blocks paths that were determined to be infeasible during the feasibility checks. Once the user is close enough to the handle (defined by a distance to $\mathbf{T}_{e2}^{e1}$), the transition to state 2 happens. Further states complete the grasp and open the drawer, as explained in detail in Section 4.3. **Credits:** This figure was taken from [38]. © 2024 IEEE.

***refinement:*** If there is one path, find further paths that do not violate kinematic limitations. Use them to compute $\mathbf{B}^w_{\text{ref}}$ for the refined TSR $\mathcal{S}^*_{\text{ref}}$. Then, instantiate an SCT with the refined TSR $\mathcal{S}^*_{\text{ref}}$ and activate the refined SCT to provide shared-control support. These two parts are described in subsections A and B of this section, respectively.

### 4.2.1 Feasibility Checking

The aim of this part of the algorithm is to find one path (in simulation) that does not violate kinematic limitations.

**Notation:** To describe the sub-steps of this part of the algorithm, we will relax the notation and refer to the poses

$$\mathbf{T}^0_{e_i^1} \dots \mathbf{T}^0_{e_i^n}$$

(i.e., the EE poses at the different via-points $1 \dots n$ of the task, and with respect to the origin, where $i$ refers to a sample $\boldsymbol{b}'_i \sim \mathbf{B}^w$) as:

$$e_i^1 \dots e_i^n.$$

Note that a few values of $i$ are relevant for us within this section. For instance, if it takes a specific value $i = \text{entry}$ (explained below), $e^1_{\text{entry}}$ is the pose at the first via-point for this TSR

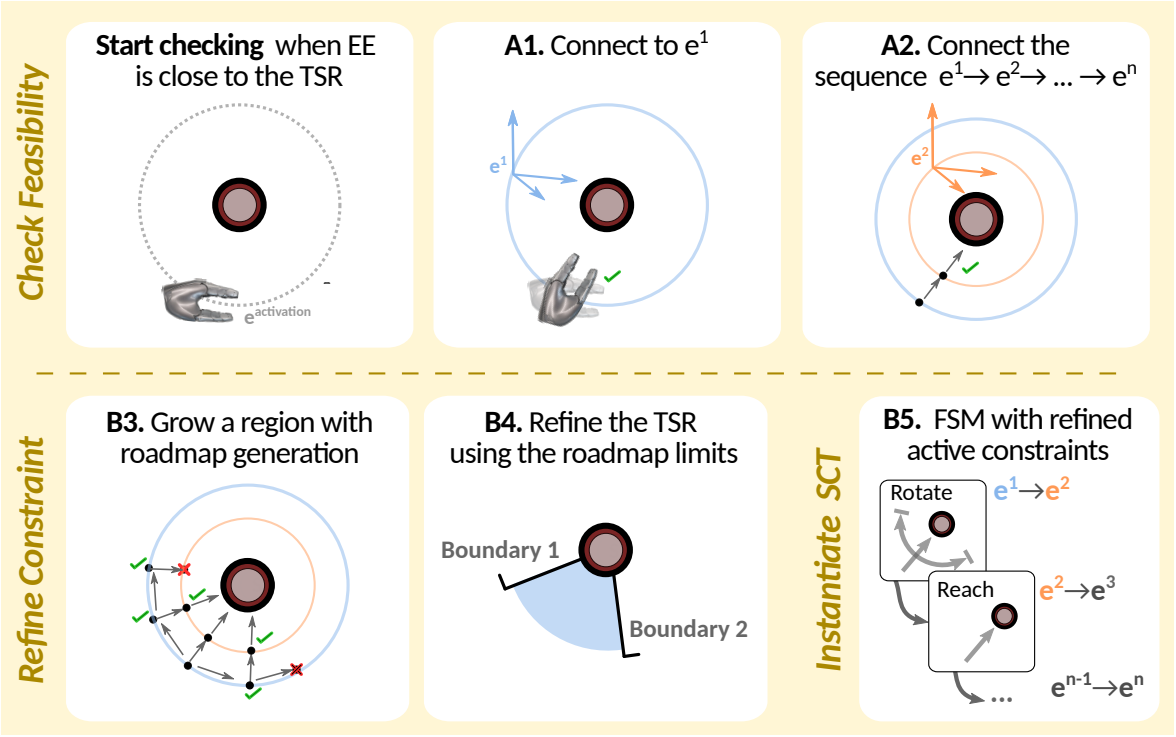



**Figure 4.7:** Illustration of the algorithm for feasibility checking, and constraint refinement. This algorithm runs in simulation before an SCT is activated. **Credits:** This figure was taken from [38]. © 2024 IEEE.

sample. For the sake of clarity, I emphasize that $e^1_{\text{entry}}$ does not refer to "the EE pose from the entry as seen from coordinate axis 1", as could be understood if compared to the $\mathbf{T}^A_B$ notation.

#### Initialization:

As a default control mode (see Section 2.3.5) the user controls either the EE translations or the EE rotations. Whenever the EE is close within a certain distance of the object (e.g., in the vicinity of a TSR first offset sample $e^1_i$), the feasibility checks algorithm is started. It is initialized with the current EE pose, i.e., not with that TSR sample. Thus, we denote this *activation pose* as $e^{\text{activation}}_{\text{entry}}$.

#### Step A1:

First, $e^{\text{activation}}_{\text{entry}}$ (the activation pose) and $e^1_{\text{entry}}$ (the closest pose respecting the constraints in the TSR $\mathcal{S}^*$) are connected. In the running example, for instance, the hand will align towards the object; see "A1" in Fig. 4.7.

In this framework, "connecting" means finding paths between any $e^j_i$ and $e^{j+1}_i$ in both EE and joint space, see the explanation after step A2. If these paths violate kinematic limitations (collisions, out-of-reach, etc.), the feasibility check fails.

**Step A2:**

This step asserts that there is *one* solution to the task, e.g., a path from the aligned hand to the object; see "A2" in Fig. 4.7. We use Eq. (4.8) to compute all poses $e^2_{\text{entry}} \dots e^n_{\text{entry}}$ from $e^1_{\text{entry}}$. Then, we connect subsequent poses, first $e^1_{\text{entry}}$ to $e^2_{\text{entry}}$, then $e^j_{\text{entry}}$ to $e^{j+1}_{\text{entry}}$ until $e^n_{\text{entry}}$.

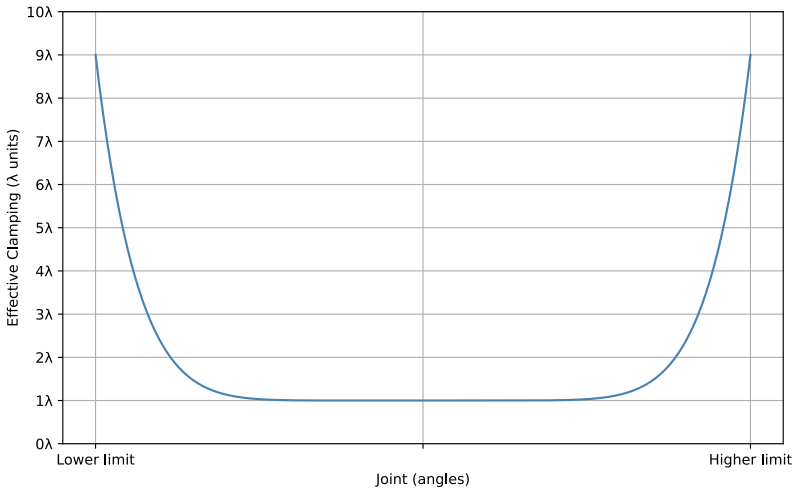



**Figure 4.8:** Effective clamping parameter on a joint as a function of proximity to a joint limit, based on [136]. If the joint is far away from a joint limit, the clamping parameter remains as the default, but increases towards either of the joint limits.

**Connecting and checking:** The EE path is generated with spherical linear interpolation, as used in SCTs [155, 199] (mentioned in Section 2.3). This is to ensure consistency of the paths generated on the robot *during* shared control, and those generated *beforehand* in simulation for the feasibility checks.

We then perform a feasibility check on the corresponding path in joint space, which is computed from the EE pose path by solving the inverse kinematics locally using the Jacobian pseudo-inverse. We use Damped Least Squares to prevent numerical instability near Jacobian singularities, by iteratively applying [34]:

$$\delta q = \mathbf{J}^\intercal(\mathbf{J}\mathbf{J}^\intercal + \lambda^2\mathbf{I})^{-1}\boldsymbol{e}, \tag{4.10}$$

where $\mathbf{J}$ is the Jacobian of the robot, $\boldsymbol{e}$ is the error between the target pose and the original pose, and $\lambda > 0$ is a damping parameter that prevents unstable behavior when near singularities. To increase the likelihood of finding a joint-space path, we further add joint limit avoidance as a secondary target by using a method from Na et.al. [136] that further clamps the joints that are closest to joint limits, i.e., by increasing the damping parameter as a function of joint limit proximity, illustrated in Fig. 4.8. In summarized terms, by clamping those joints, this method exploits the redundancy of the kinematic chain (2 extra joints in the

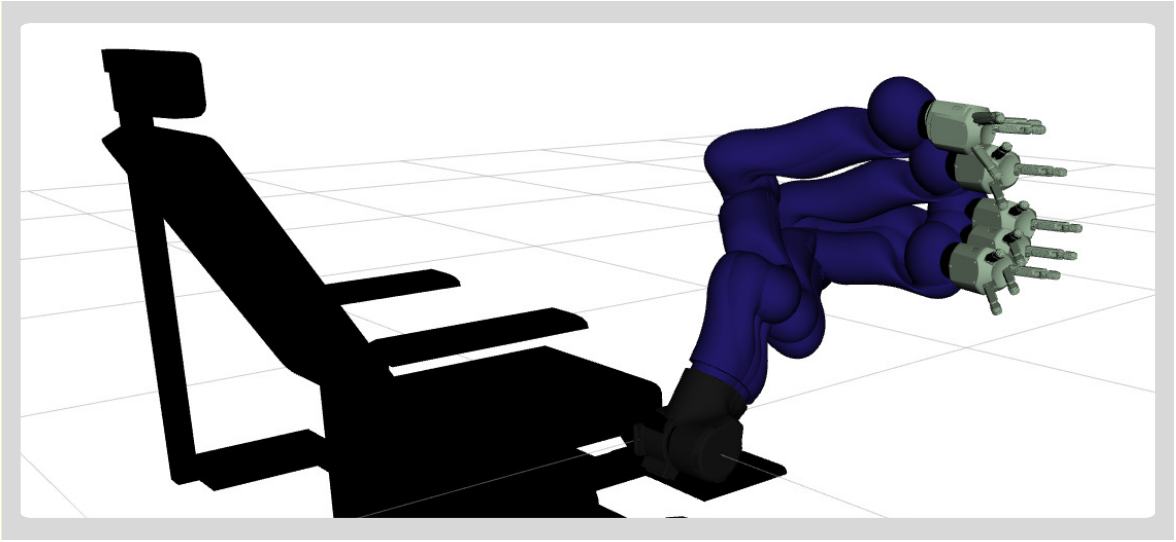


**Figure 4.9:** An example of an inverse kinematics solution on EDAN using Damped Least Squares, to move the EE from the top to the bottom pose.

case of EDAN) to find kinematic solutions that avoid the joint limits. The rationale for this is trying to maximize the chances that a path is found, if it exists, and entails the assumption that the robot's controller is able to avoid the joint limits in a similar way. An example of a path resulting from this method is illustrated in Fig. 4.9.

**Feasible or infeasible tasks:** Using Eq. (4.10), we check if a resulting joint-space path exists, or if it leads to kinematic limitations, including obstacles. We define a task as infeasible if either Step A1 or A2 fails, i.e., if a simple and local path that solves the task does not exist. If this happens, the SCT is not activated for the user, and steps B3, B4, and B5 are not executed. This scenario is further discussed in Section 4.2.3, presenting an algorithm for resolving infeasible tasks using a motion planner.

**Collision checking with obstacles or the robot body:** For collision checking, we use the FCL library [143] and a 3D model of the robot with all its links (not just the EE). FCL supports obstacles in the form of primitive shapes (e.g., a box), a 3D mesh, or an Octomap [81] volume representation. The former two were used in the experiments (e.g., in Fig. 4.12 of Section 4.2.2), while the latter is shown in Fig. 4.10 for illustration purposes. Our method also models expected collisions by using the CAT representation introduced in Chapter 5 (e.g., the contact between the fingers and an object being grasped).

**Illustration:** Feasibility checks are illustrated in Fig. 4.11. In the starting position depicted, SCTs would activate shared control, as the EE is close to the bottle. However, this is not intuitive: the shelf sideboard is an obstacle between the bottle and the shelf. The feasibility checks identify this and do not activate shared control. To activate shared control, the user

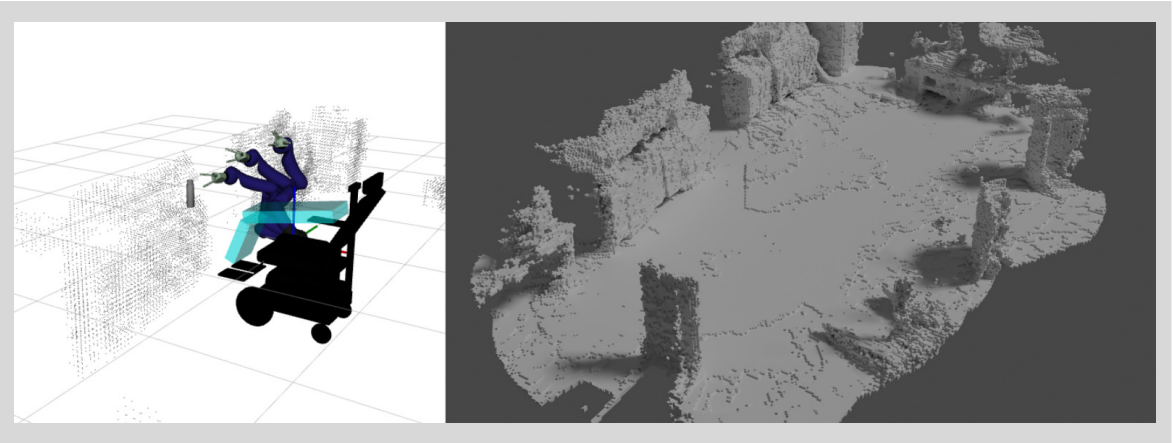


**Figure 4.10:** The collision checking library in our framework supports Octomap obstacles, shown for a grasping task on EDAN (left). In this example, the 3D occupancy grid on the right (a snapshot from our lab) is used as an obstacle for the joint-space path generation on the left. **Credits**: Special thanks to Dominik Winkelbauer and Valentina Choquenaira for their assistance with implementing Octomaps on EDAN. The occupancy grid on the right was recorded by Dominik.

would first navigate the EE to the compartment where the EE is.

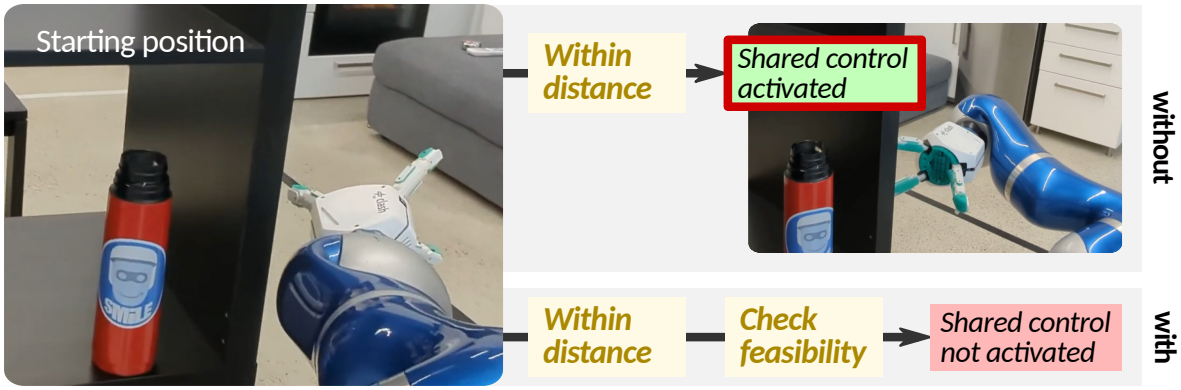



**Figure 4.11:** Illustration of the difference between shared-control activation without (top) and with (bottom) feasibility checking. **Credits:** This figure was taken from [38]. © 2024 IEEE.

### 4.2.2 Constraint refinement and SCT instantiation

If the task is deemed feasible in Part A of the algorithm, its result is one path that connects $e^{\text{activation}}_{\text{entry}}$ and $e^{n}_{\text{entry}}$ whilst respecting kinematic violations. As the constraints in TSRs and SCTs are regions rather than paths, the next step is to determine the appropriate ranges for the region in the TSR by exploring further feasible paths from the one feasible path determined in Part A of the algorithm. In Step B5 of the algorithm, the refined regions in the TSR are converted to refined constraints in the SCT. Hence, this process is called 'constraint refinement', rather than 'region refinement'.

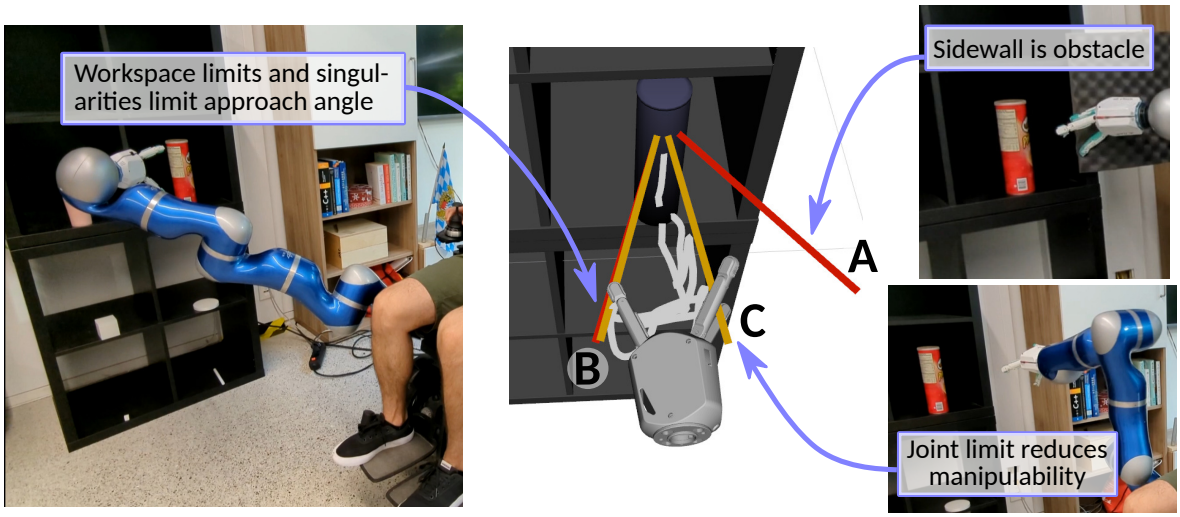



**Figure 4.12:** The photos left and right show kinematic limitations that the user may run into without refined constraints. They correspond to task runs where the refined constraints were ablated, for illustration purposes. The refined constraints that would preclude these limitations are highlighted by the straight lines projecting from the cylindrical object in the middle. The constraint labeled **A** arises from the failure of paths colliding with the sidewall (top-right photo where the fingers touch the shelf). **B** arises from workspace constraints, singularities, and the left sidewall (left photo). **C** arises from a specific initial position where the robot is close to a joint limit that reduces its manipulability if gone beyond the line (bottom-right photo). The white lines between the constraints B and C show EE trajectories executed by the user, highlighting that there is freedom of movement between the constraints. **Credits:** This figure was modified from [38]. © 2024 IEEE.

#### Step B3:

We grow a roadmap from the initial path, see step B3 in Fig. 4.7. We first retrieve the vector $\boldsymbol{b}'_{\text{entry}}$, containing the position and Euler-angle orientation of the entry pose $e^1_{\text{entry}}$ expressed in the coordinates of $w$ (see Eq. (4.3)). Then, we obtain a vector $\boldsymbol{b}'_i$ in the vicinity of $\boldsymbol{b}'_{\text{entry}}$, within the bounds of $\mathbf{B}^w$. We perform two types of connections: first, we connect the first offset in $\boldsymbol{b}'_{\text{entry}}$ (i.e., $e^1_{\text{entry}}$) to the first offset in $\boldsymbol{b}'_i$ (i.e, $e^1_i$). Second, we connect all offsets $e^1_i \dots e^n_i$, akin to Step A2. If both types of connections are successful, we mark vector $\boldsymbol{b}'_i$ as feasible, and infeasible otherwise.

If $\boldsymbol{b}'_i$ is feasible, we then keep growing the roadmap by obtaining a new vector in its vicinity and repeating the connections, continuing the procedure iteratively until finding kinematic issues, obstacles, or heuristics, such as a maximum TSR size.

#### Step B4:

This step refines $\mathbf{B}^w$ into $\mathbf{B}^w_{\text{ref}}$ using the roadmap. $\mathbf{B}^w_{\text{ref}}$ is the matrix containing the largest limits (min and max) that surrounds the set of feasible vectors $\boldsymbol{b}'_i$ in the roadmap, but such that it does not contain any infeasible vector.

**Step B5:**

In the final step, an SCT is instantiated with the refined TSR with the procedure defined in Section 4.1.4. The ACs of the instantiated SCT are refined, i.e., the constraint does not include the simulated paths that were not feasible.

**Illustration:** Constraint refinement with a shelf obstacle is shown in Fig. 4.12, displaying real-world scenarios with EDAN where the constraints are ablated. Possible kinematic limitations in the example include damaging the fingers with the sidewall obstacle, running into singularities, or losing manipulability due to a joint limit.

### 4.2.3 Motion planning for enabling infeasible tasks

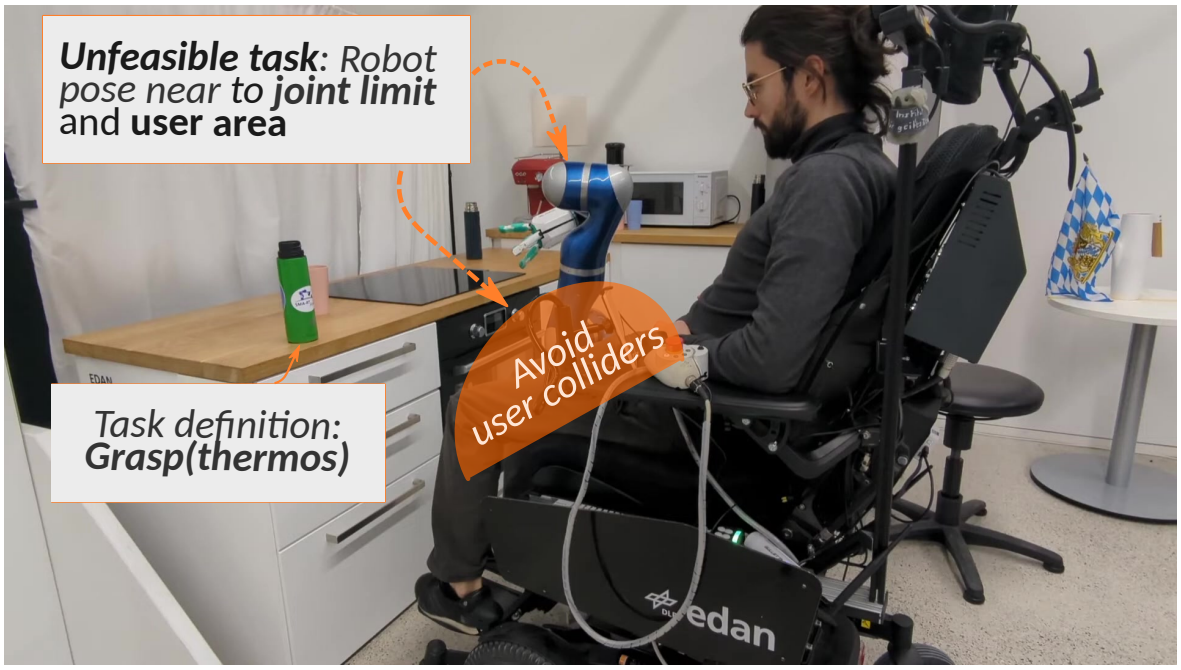



**Figure 4.13:** An infeasible thermos grasping task in the proximity of obstacles and joint limits. The robot does not find a local path that leads the EE in the task direction, whilst e.g., avoiding the virtual walls that protect the user (depicted here as "user colliders"). © 2025 The thesis author.

The method presented in this chapter enables the robot to check the feasibility of a task before it is started, thus preventing the user from ascertaining the feasibility of a task during execution. But it relies on finding local kinematic solutions to a task. This approach works if "simple" paths exist that solve the task, and preclude its execution if they do not. An infeasible task example is shown in Fig. 4.13.

To enable users to still execute tasks that are flagged as infeasible, we extend the method from Sections 4.2.1 and 4.2.2 with a motion planning algorithm that finds a task solution globally. This is achieved by moving the joints to a position where the task is feasible again,

which is colloquially known as "*reconfiguring*" the robot.

The algorithm consists of four steps. As it is independent from the feasibility checking algorithm in Sections 4.2.1 and 4.2.2, I refrain from the **A-B** nomenclature from the previous sections and simply list the steps. These are:

1. We find a TSR-first-offset $e_i^1$ to enter the task. We use a global sampling-based motion planner (a Constrained Rapidly-Exploring Random Tree, the uni-directional variant of the so-called CBiRRT [22, 114] algorithm) to find a collision-free joint-space path from the robot's initial state to a configuration where the EE is at the TSR first offset, using the TSR to define the path goal[3].

2. We then use a so-called shortcut method [62] to improve the quality of the motion plan by minimizing its length and increasing its distance to obstacles. Let $q_{\text{reset}}$ be the final configuration of this motion plan, and $e^1_{\text{entry-candidate}}$its corresponding EE position. The resulting motion plan is not yet executed on the robot.

3. We conduct the feasibility checks (Steps **A1**-**A2** from Section 4.2.1) as if the user had activated the SCT on $e^1_{\text{entry-candidate}}$, using $q_{\text{reset}}$ as the starting joint configuration. If successful, we conduct the constraint refinement procedure as well (Steps **B3**-**B5** in Section 4.2.2).

4. We execute the joint-space motion plan autonomously, ending in $q_{\text{reset}}$ with the EE in $e^1_{\text{entry-candidate}}$. From there, the user can complete the task in shared control.

If the feasibility checks on Step 3 fail (i.e., the task is not feasible if the robot would be at $q_{\text{reset}}$ at the start of the shared control task), the algorithm can be repeated until a solution is found. This is one of the topics of the discussion in Section 4.4.

**Illustration:** A motion plan that solves the previously-infeasible task in Fig. 4.13 is shown in Fig. 4.14, displaying a joint space motion executed autonomously on EDAN. The arm is brought to a new start configuration from which the task is executed in shared control.

## 4.3 Experimental evaluation

In [38], my colleagues and I presented the set of experiments reported here, intended to validate our feasibility checks framework. The EDAN robot with a three-fingered hand was used (see Section 2.4). The research questions that we aimed to answer were:

[3] I note that the first step can be replaced by sampling directly $e^1_{\text{entry-candidate}}$from the TSR, and then using a conventional rapidly-exploring random tree (RRT [110]) to compute a collision-free joint-space motion towards it.

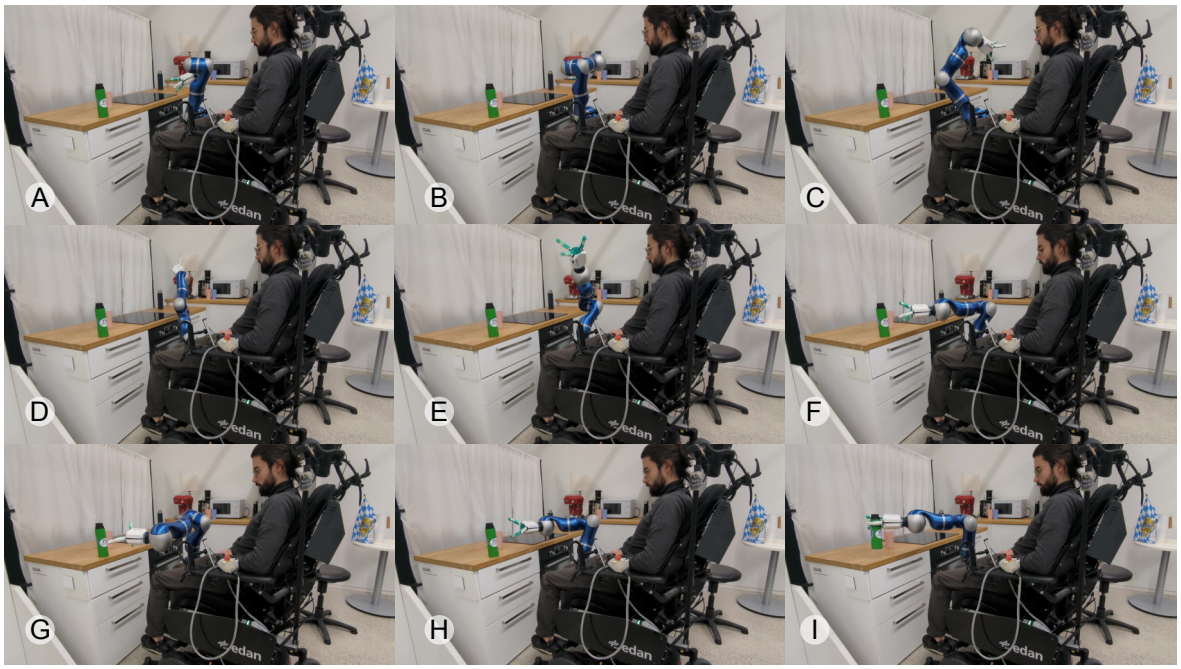



**Figure 4.14:** The robot finds a global motion plan to solve the previously-infeasible task in Fig. 4.13, starting from (**A**). A reconfiguration (**B**-**F**) brings the robot to $q_{\text{reset}}$ , a joint space position where the task is feasible again (**F**). Starting there, the user is free to explore the task region within the refined constraints (**G**-**H**). Finally, the user grasps the thermos successfully (**I**). © 2025 The thesis author..

- To what extent can our feasibility checks preclude activation of SCTs that are infeasible? Are there many false positives?

- How fast is the computation time of feasibility checks in the context of EDAN?

#### Collision models

During normal EDAN operation, a set of reactive virtual walls prevents the robot from getting close to the user. Thus, the collision models (a.k.a. *colliders*) used in the experiment included both the robot and those virtual walls. As noted on each experiment, either the real robot or the digital twin with a full dynamic simulation (including the controller, see [195] for more information) was used.

#### Interface

Whenever run in shared control, a 3D joystick interface was used to control the robot EE, which was commanded by one of the authors. When noted in some of the experiments below, and to prevent bias, the EE was commanded instead by the Automaton algorithm that automatically completes an SCT task by issuing joystick commands.

#### Evaluation Tasks

Three tasks – highlighted in Fig. 4.15 – were used in the evaluation: `Container`: grasping a tall, cylindrical container; `Mug`: grasping a mug, i.e., a short and cylindrical container; `Drawer`: grasping a drawer's handle and opening it. To test the feasibility checking, we intentionally added sources of kinematic limitations, often leading to task failure, and described on each experiment.

The TSRs for all tasks had EE offsets for *entry* ($\mathbf{T}_{e^1}^{w'}$), *pre-grasping* ($\mathbf{T}_{e^2}^{e^1}$) and *grasping* ($\mathbf{T}_{e^3}^{e^2}$), shown in Fig. 4.4. The container tasks (`Mug` and `Container`) had also a *post-grasp* transform above the object ($\mathbf{T}_{e^4}^{e^3}$). The task `Drawer` had two variations: opening a drawer with a horizontal handle (`Drawer-H`), containing an EE offsets where the drawer *is fully open and the hand releases* ($\mathbf{T}_{e^4}^{e^3}$), and another *post-release* one near the handle ($\mathbf{T}_{e^5}^{e^4}$); and a more general variant for pulling a vertical bar (`Drawer-V`) with only a *post-grasp* offset, where the object is pulled ($\mathbf{T}_{e^4}^{e^3}$).

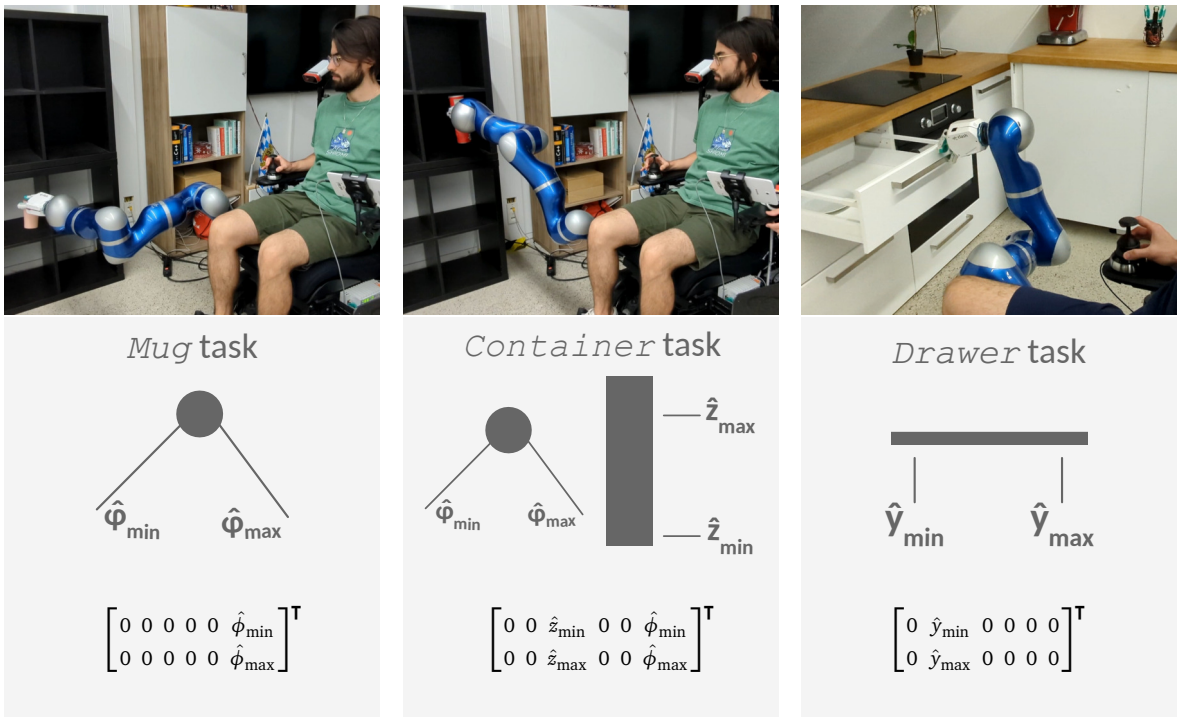



**Figure 4.15:** The three TSRs used in the evaluation. For each task, a schematic is shown containing the TSR, the matrix $\mathbf{B}^w$, and a snapshot of a completed task in the real robot. **Credits:** This figure was modified from [38]. © 2024 IEEE.

### 4.3.1 Appropriate SCT activation (simulation)

To what extent can our feasibility checks preclude activation of SCTs that are infeasible? 447 trials of `Drawer-V` were run in the dynamic simulation, using a vertical bar with no mass or collision meshes as a dummy target. The initial position of the bar varied in the $yz$ plane

with fixed $x$ values (we aggregated results for $x = 0$m & $x = -0.15$m).

On each trial, an SCT was activated with an EE position sampled uniformly in the bounds of the TSR first offset (TSR max length: 0.3m), and an EE orientation was sampled uniformly from a fixed mean with pre-coded tolerances (±0.4 roll (r), ±1.15 pitch (p) ±1.5 yaw (y) rad;); a corresponding robot configuration was sampled using conventional inverse kinematics. Ground-truth trials were completed by the Automaton [36]. A trial would be marked as *failed* if the robot would have a self-collision or if the EE could not reach the transitions frames projected from the active constraints using pre-coded tolerances (±0.02m, 0.15rad), or *successful* otherwise.

In parallel to SCT activation, the feasibility checking algorithm (Step A1, A2) was run to predict whether the task would succeed or not. Fig. 4.16 and Table 4.1 summarize the prediction metrics of the feasibility checks as a function of the bar location. We report **accuracy** as the ratio between correct feasibility predictions and total feasibility predictions; **recall** as the ratio between the number of trials correctly predicted to fail and the total trials that failed; and **precision** as the ratio between trials correctly predicted to fail and the total trials predicted to fail. It should be noted that these metrics correspond to feasibility prediction, not task success.

**Sources of kinematic limitations:** low manipulability in the selected workspace due to (i) proximity of the EE to the robot origin and virtual walls (e.g. in $y = 0.05$m, $z = 0.4$m) and (ii) reachability out of the workspace (e.g. $y = 0.65$m or $z = 1$m); additionally: aleatory proximity to joint limits due to randomly sampled start configurations.

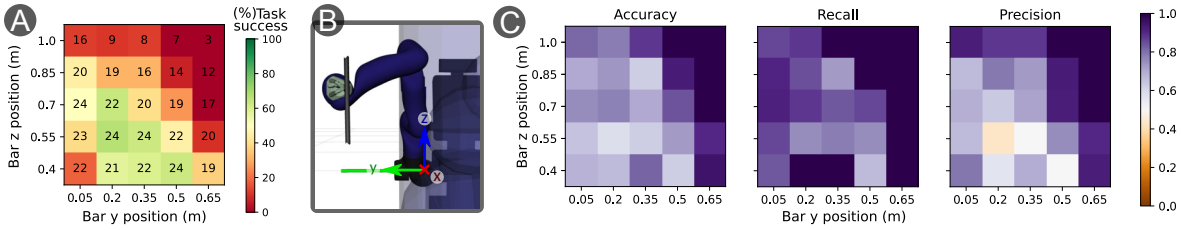



**Figure 4.16:** **A.** Ground truth task success. The numbers in boxes indicate the number of trials run at each bar position in $yz$. The color signifies task success (%). **B.** A snapshot of the robot simulation and the dummy target in a task. **C.** Evaluation of feasibility predictions as a function of the bar position in $yz$ with respect to the arm origin (coordinate frame in **B**). **Credits:** This figure was taken from [38]. © 2024 IEEE.

### 4.3.2 Appropriate SCT activation (real robot)

20 trials of the `Mug` task were run on the real robot. The robot was initialized on an initial fixed seed configuration. The mug position was fixed. The trials were started by the experimenter driving the EE to an SCT activation pose nearby the mug, with varying EE positions (but not orientations), and were completed by the Automaton [36]. As ground truth, a trial would be marked as *successful* if the robot would grasp and lift the mug, regardless of any minor self-collision, and *failed* otherwise. In parallel to SCT activation, the feasibility check-

ing algorithm (Step A1, A2) was run to predict task failure. Fig. 4.17 illustrates the ground truth and predicted failures and successes. Table 4.1 summarizes these results.

**Sources of kinematic limitations:** 10/20 trials were seeded from a configuration between a joint limit and the virtual walls, yielding low manipulability in the task direction. The other 10 were from a configuration with high manipulability in the task direction.

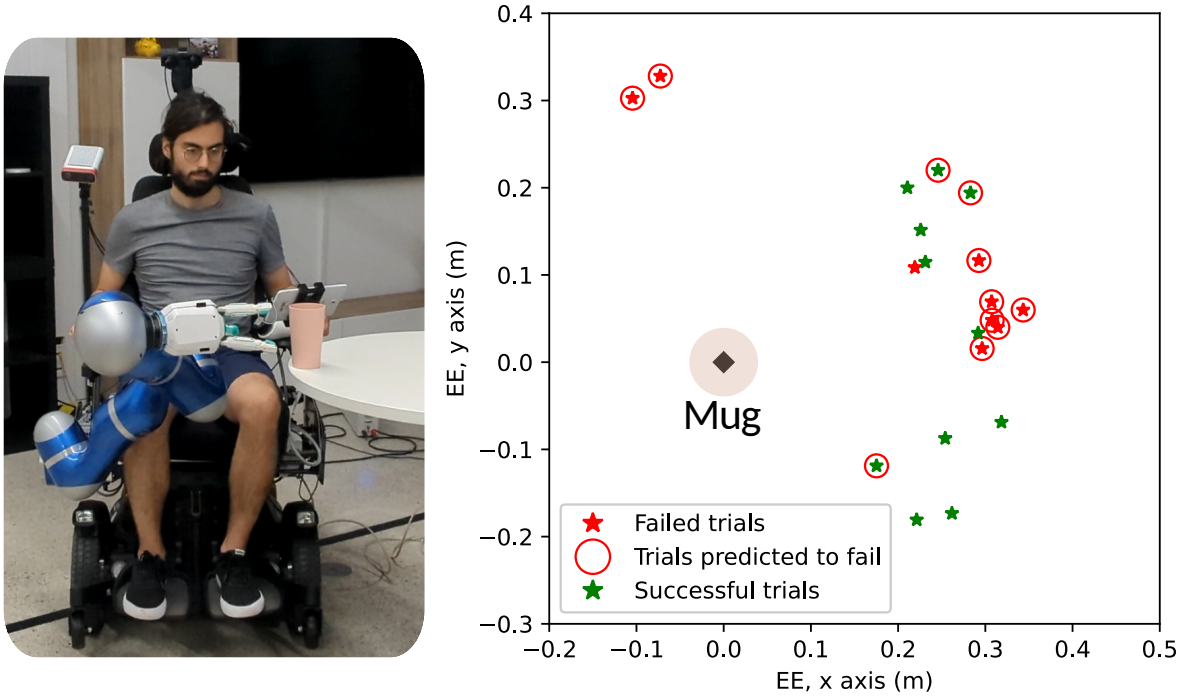



**Figure 4.17:** Summary of the results on the real robot. Left: snapshot of the experimental setup. Right: success as a function of EE activation positions. **Credits:** This figure was taken from [38]. © 2024 IEEE.

| | | **Feasibility checks prediction** | | |
|---|---|---|---|---|
| Experiment | Trials(failed) | Accuracy | Recall | Precision |
| Simulation | 447(280) | 0.76 | 0.89 | 0.76 |
| Real robot | 20(9) | 0.80 | 0.89 | 0.73 |

**Table 4.1:** Effectiveness of infeasibility prediction. **Credits:** This table was taken from [38]. © 2024 IEEE.

### 4.3.3 Computation times benchmark

To benchmark the computation time for the feasibility checks and constraint refinement, Steps A1-B5 were run for different robot configurations, 20 in which the task was infeasible, and 20 in which it was feasible. An EE position was sampled uniformly in the bounds of the TSR (using the first offset), and an EE orientation was sampled uniformly from pre-coded boundaries (`Mug` & `Container`: r±0.8 p±0.2 y±0.3rad, `Drawer-H`: r±0.3 p±1.58 y0.3±rad); a corresponding robot configuration was obtained using conventional inverse kinematics. It

should be noted that no SCT (step B5) was instantiated for infeasible tasks. The experiment was conducted with a 3.70GHz CPU. Table 6.2 summarizes the computation times for the three tasks.

| | Mug | Container | Drawer-H |
|---|---|---|---|
| Steps A1-A2 (⇒infeas.) | $0.43 \pm 0.16$ | $0.63 \pm 0.45$ | $0.48 \pm 0.31$ |
| Steps A1-B4 (⇒feas.) | $1.55 \pm 0.40$ | $2.86 \pm 1.04$ | $2.17 \pm 0.55$ |
| Step B5 (⇒feas.) | $0.07 \pm 0.03$ | $0.07 \pm 0.04$ | $0.08 \pm 0.03$ |

**Table 4.2:** Computation ($s$), $\mu \pm \sigma$ over 20 trials. **Credits:** This table was taken from [38]. © 2024 IEEE.

### 4.3.4 Discussion of the experimental results

The results in Fig. 4.17 and Table 4.1 show that the feasibility checking framework has high recall (0.89 in Table 4.1), and thus it can successfully prevent activation and instantiation of infeasible SCTs. In the real robot experiment, all but one of the overall failed trials (8 of 9) were successfully flagged by the feasibility checks as infeasible. This means the user is averted from ascertaining the infeasibility of these tasks through trial-and-error.

The method does suffer from a number of false positives in some regions of the workspace, i.e., deeming a trial infeasible, even though it is not. This results in relatively low aggregated values of precision $(0.73, 0.76)$ and accuracy $(0.76, 0.8)$ in Table 4.1. In Fig. 4.17, for instance, 3 out of 20 trials were predicted to fail when they did not. In the simulated experiment, this is most pronounced in the area of high task success $y = 0.2, z = 0.5$ (right graph in Fig. 4.16), where the robot has large reachability. We believe that this is due to differences between our kinematic model for connecting via-points and the dynamic controller of the robot, since the latter can dynamically exploit the robot's nullspace, and find task solutions more often.

In order to reduce experimental bias during a trial, the runs of the experiment were completed with the automaton algorithm presented in Chapter 3. This highlights a feature of our cognitive shared control system: precisely that it generates feasibility predictions for a task regardless of which control mode is selected by the user. It would have been entirely possible for the experimenter to complete the task in shared control, and therefore, the system provides the user with combined flexibility and robustness.

Fig. 4.12 shows how the refined constraints algorithm incorporates different kinematic limitations due to obstacles (the shelf sideboard), singularities, and joint limits. While a skilled user may not need constraints to avoid the shelf, it may be useful for novice users and for users with noisy interfaces such as electromyography [196]. Also, in situations such as Figs. 1.5 and 4.17, task failure is not immediately apparent, because joint limits and virtual walls are invisible to the user. In the figure example, the user fails to grasp a mug in shared control after he robot's initial configuration was near kinematic limitations due to a joint limit and the virtual wall, even though such a situation may not be immediately apparent to the

user. Thus, our hypothesis is that feasibility checks can help skilled users as well.

Finally, from Table 6.2, we see that determining if a task is infeasible took between 0.43-0.63 seconds on average. Infeasible tasks can thus be excluded quickly. Instantiating the SCT was also very fast, i.e., less than 100ms. The constraint refinement in B3/B4 takes the longest, as multiple paths must be computed. The scope for this task benchmark was only the feasibility checks and constraint refinement algorithms (Steps A1-B5), and thus this time benchmark did not include the algorithm for motion planning from Section 4.2.3. As sampling-based motion planning is a more costly algorithm, it would be expected that it would take longer.

## 4.4 Limitations and outlook

This chapter introduces a framework for feasibility checks in the context of shared control tasks, and shows that it can be applied off-the-shelf for supervised autonomy situations as well. Thus, the combined frameworks of Chapters 3 and 4 provide the desired features of **Flexibility** and **Robustness**. There are several directions where this work can evolve in future work, which are highlighted next:

### 4.4.1 Faster feasibility checks

An avenue for improvement is to explore whether the feasibility checks algorithm can be made *faster* by exploiting lookup in pre-existing kinematic models of the robot, such as the so-called reachability maps [149]. Here, involving heuristics can also help speed up, such as initializing the shared control task with a single path computed in Step A1-A2, to be then replaced with the entire region soon afterwards.

Another idea is increasing the algorithm accuracy by using both dynamic and kinematic models in the connection step. This is akin to previous work on assembly tasks [86], where feasibility checks are run first on a fast kinematic layer and only if necessary on a more expensive dynamic one.

### 4.4.2 Global feasibility checks

The connection algorithm shown in Fig. 4.7 explores the task region locally from the activation point. This is a design choice, considering there is a human in the loop, who provides task knowledge, who may not be unnecessarily contradicted. However, the locality of the algorithm may lead to sub-optimal search spaces for the task in some cases, yielding small refined TSRs, or none. As an example, the TSR with the refined constraints BC in Fig. 4.12 could be converted into BA, by reconfiguring the robot into a new starting position.

One could obtain wider solution spaces if one searched globally, e.g., bidirectionally between the goal and the start. However, this contrasts with the affordance-based nature of constraints in shared control in the related work [155, 193, 93, 180]. Future work can focus on intelligently selecting between local and global solutions depending on the user's interest and confidence.

The approach from Section 4.2.3 provides an algorithm and a proof of concept for global feasibility checks in the event of infeasible tasks. One issue with the presented algorithm is that it needs to iterate the feasibility checks until a solution is found, which can be costly. Such an approach can benefit from more principled *backtracking* algorithms (see [117] for an example) that represent a search tree with modifiable task parameters, pruning it if solutions fail. We leave the study and application of such backtracking techniques to future work.

### 4.4.3 Dynamic obstacles

Finally, the approach evaluated here checks kinematic feasibility before task execution. But it does not take dynamic obstacles into account. To do so, it could be combined with methods that do [193, 5, 177, 176, 169]. An advantage of a combined method (feasibility checking and reactive checking together) is that reactive checking alone during the task does not guarantee the existence of task solutions, e.g., the user may still guide the robot EE into configurations that preclude task completion, for instance, due to competing constraints.

# 5

# Task planning for shared control with Constraint-Action Templates

*This chapter considers user goals for which it is necessary to sequence and execute multiple SCTs over time. The objective of this chapter is to provide a framework for goal-guidance in the context of shared control action. To facilitate planning with shared control actions, this chapter introduces an action representation called Constraint-Action Templates (CATS), which combines the advantages of Action Templates [117] and Shared Control Templates [155]. The CATs action representation and the derived planning framework generate plans that can be executed with both shared control and supervised autonomy, even switching between them during task execution.*

***This chapter originates from a paper for the 2022 International Conference on Robotics and Automation (ICRA), and I include some literal text portions and figures. However, it has been reworked to fit as a chapter within the thesis concept, and also features a few extensions such as Section 5.1.1 and some figures and listings. See: [37]. Copyright note in the footnote [1]. Listing 5 was taken from [115].***

So far, we have assumed that SCTs are activated either because they are selected by the user or inferred from the available objects in the scene [196, 17]. This chapter instead considers goals for which it is necessary to *sequence* and execute multiple SCTs over time. To facilitate the task planning required for this sequencing, we propose the Constraint-Action Templates, an action representation that complements the SCTs — enabling shared control — with the symbolic knowledge required to plan tasks, inspired by the Action Templates

[1] Whenever parts of the manuscript are taken literally: © 2024 IEEE. Reprinted, with permission from the senior author, from S. Bustamante, G. Quere, D. Leidner, J. Vogel, and F. Stulp, CATs: Task Planning for Shared Control of Assistive Robots with Variable Autonomy, International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022

(ATs) [116]. The resulting planning and execution framework based on CATs enables cognitive shared control to provide guidance towards goals and consider longer time horizons than is possible with single SCTs.

An important feature of CATs is that it endows the task plan with freedom of movement, to empower the user to be in control of the planned actions following the principles from Fig. 1.2 (as illustrated in Fig. 1.6).

This chapter is divided as follows: It first provides a background on symbolic planning in Section 5.1, including the ATs. After that, it introduces the novel action representation, the CATs in Section 5.2, and presents an overview of how they are used for planning and execution in Section 5.3, including experiments on EDAN in Section 5.4. The connection to the feasibility checks is mentioned briefly in Section 5.2.5.

## 5.1 Background: the Planning Domain Description Language

I first provide (in Section 5.1.1) a primer in symbolic planning in the broad context of the PDDL language, which is used extensively in many artificial intelligence (AI) planning methods, including the thesis. Afterwards, I describe the Action Templates, a representation from our research institute that combines symbolic and geometric planning in robotics, serving as inspiration and precedent for the CATs representation introduced later.

### 5.1.1 Symbolic planning with PDDL

**Listing 1** A PDDL action example, for grasping an object. The action involves a robot and an object. The prerequisite is that the object is free; the effect is that the object is grasped.

```
;; Parameters:
;; what are the entities involved in the action?
:parameters (?o - _object ?r  - _robot)

;; Preconditions:
;; what is the necessary world symbolic state for this action to be scheduled
  ?
:precondition (and(free ?r))

;; Effect:
;; what elements of the world symbolic state change during the action?
:effect  (and(grasped ?o  ?r)  (not (free ?r)))
```

The AI planning community has been working on symbolic planning for a long time, with a significant milestone in the STRIPS (Stanford Research Institute Problem Solver) planner

from 1971 [108]. A derived language used to represent "*planable*" actions is the Planning Domain Definition Language, or PDDL. PDDL is a widely used language with numerous features; therefore, I will only summarize the elements relevant to the thesis.

A PDDL plan creates a sequence of actions to achieve a goal, based on a fixed schema about the actions. An action looks like Listing 1, and defines the expected changes to the world state if the robot executes this action.

**Listing 2** A PDDL action example, for opening the fridge

```
;; Open fridge action
:parameters (?f - _fridge ?r  - _robot)
:precondition (and(free ?r)  (not (opened ?f)))
:effect  (and (opened ?f))
```

**Listing 3** A PDDL action domain example including pick and place actions.

```
(define
 ;; Specify a domain name
 (domain grasping)

 ;; What predicates will be used by actions in this domain?
 (:predicates
  (free ?r - _robot)
  (grasped ?o - _object ?r - _robot)
  (on ?o - _object ?s - _surface)
  ;; no need for (_opened ?f - _fridge)
 )

 ;; Grasping action
 (:action GRASP
  :parameters (?o - _object, ?r  - _robot)
  :precondition (and(free ?r)  (not (grasped ?o ?r)))
  :effect  (and(grasped ?o  ?r)  (not (free ?r)))
 )

 ;; Placing action
 (:action PLACE_ON_SURFACE
  :parameters (?o - _object , ?r  - _robot, ?s - _surface)
  :precondition (and(grasped ?o ?r))
  :effect  (and(not (grasped ?o ?r))  (on ?o ?s) (free ?r))
 )
)
```

As introduced in Section 2.2, these changes apply to *predicates*, i.e., statements of truth in first-order logic. For example, `(opened ?fridge)` literally means "*it is True that the fridge is opened*", whereas `(free ?r)` means that the robot (end-effector) is free. Thus, any action that requires an open fridge will formally require `(opened ?fridge)` as a precondition. Likewise, any action that results in an open fridge will specify it as its effect. Intuitively, the action

of opening or closing the fridge changes the truth value of this statement, as illustrated in Listing 2. Finally, it is worth noting that we use a **closed-world assumption**, meaning that a semantic state that is not specified as True is considered False. If `(opened ?fridge)` is not stated (e.g. in a world model), it consequently implies that "*it is not True that the fridge is opened*".

Consistent with the notation of other papers, I use a trailing underscore for object classes such as `_fridge` or `_bottle`, and remove the underscore for specific object instances such as `lab_fridge` or `green_lab_thermos`. More examples are shown in Section 5.2.4. Finally, note that statements whose truth value does not change with the action are assumed to remain the same. Thus, it would be redundant to add them to the action effects: in the example, the robot hand is free (i.e., `(free ?r)`) before and after opening a fridge.

**Listing 4** A PDDL problem file, including the grasping domain from the previous listing

```
(define (problem grasp_bottle)
 (:domain grasping)

 ;; What are the relevant object instances and their classes?
 (:objects
  arm - _robot
  bottle - _object
  table - _surface
 )

 ;; What is the initial state of the world?
 (:init
  (free arm)
  (not (grasped bottle arm)
 )

 ;; What is the goal of the plan?
 (:goal
  (and (on bottle table) )
 )
)
```

**Domain file:** But cognitive agents can execute multiple actions, not just one. To specify all of them, PDDL uses a domain text file (domain.pddl, see the example in Listing 3) that specifies the schema to which the plan has access. In our case, we are interested in two representations:

- The list of actions the robot can plan with.
- The list of predicates that are used in the actions.

**Problem file:** Using the domain file in Listing 3, the robot can plan a pick-and-place action. If it grasps the object, `(grasped ?o ?r)` will be `True`. If it releases it, it will be `(free ?r)` and `(on ?o ?s)`. Thus, if the robot is not grasping an object and wants to place it on a surface, it will be guided to first grasp it, fulfilling `(grasped ?o ?r)`, which is the precondition to `PLACE_ON_SURFACE`. Then it will be able to place it.

To formalize such a planning problem, one can use a so-called problem text file (problem.pddl, see example in Listing 4). It includes the relevant object instances, the initial state of the world, and the goal of the action.

**Planner:** Finally, a symbolic planning algorithm can be used to create a plan that starts from the initial state and ends at the goal state, as specified in the `problem.pddl` file, with the actions defined in the `domain.pddl` file. There exist many methods for this. In my thesis and all related papers, we employ a method called Fast Downward, a domain-independent planning system based on heuristic search [77].

### 5.1.2 Action Templates

So far, PDDL planning can tell the robot *what it can do*, i.e., provides declarative knowledge for the robot to sequence a set of actions. But how to provide the robot with information about *how to do what it can do*, or in other words, procedural knowledge?

Our work builds on the Action Template (AT) paradigm by Leidner et. al. [115, 117], which is an action representation that endows PDDL actions with geometric information to conduct a task, thus enabling hybrid planning (recently also commonly referred to as Task and Motion Planning, TAMP). An example AT for a picking task (`_object.pick`) is shown in Listing 5.

While the header of an AT is a declarative action definition specified in PDDL, thus enabling the generation of a domain, as in Listing 3, its body is a sequence of robot operations that generate motion plans, linked to the geometric properties of the objects in the world. In other words, the robot knows the procedure of actions relative to object frames of interest (approach, grasp, etc.).

AT enables **goal-oriented planning**. The input to a planning process is a goal in PDDL, as the one described in the problem file in Listing 4, `(on bottle table)`. The planning method is called hybrid, as it combines symbolic and geometric planning methods:

- First, the *symbolic planning* algorithm retrieves the symbolic headers of all ATs, and a PDDL representation of the current state of the world. With them, it creates problem.pddl and domain.pddl files, which are readily used for planning. It then uses fast downward to generate a symbolic plan, i.e., a sequence of ATs whose execution would produce a net effect that satisfies the given goal.

**Listing 5** AT for `_object.pick`, taken from Leidner et.al. [115]. © 2012 IEEE.

```
;; Header: declarative action specification in PDDL

:parameters (?o - _object ?m  - _manipulator ?t  - _tray)
:precondition (and(free ?rn)  (on  ?o  ?t))
:effect  (and(bound ?o  ?m)  (not. (free ?m)) (not. (on  ?o  ?t.)))


;; Body: a sequence of geometric operations that generate
;; motion plans and movements

(...)
operations = [
('move_hand', manip, graspset.approach_grasp) ,
('plan_to', manip, graspset.approach_frame, object_.frame),
('plan_to', manip, graspset.grasp_frame, object_.frame),
('bind', manip, object_.name),
('move_hand', manip, graspset.pre_grasp),
('move_hand', manip, graspset.grasp)]
```

- After that, the *geometric planner* is started. It chains the geometric operations in the AT sequence, which are called in order to generate a motion plan (e.g., with a rapidly-exploring random tree, RRT [110]). The geometric planner executes the plan in simulation (e.g., with a digital twin of the robot) to assert reachability and collision avoidance.

The AT hybrid planning also enables *backtracking*: If a simulation fails, the planner re-tries with different parameter sets; if all fail, a backtracking mechanism discards the action and restarts the symbolic planner [117]. Finally, if the planner can simulate the whole plan, it is executed on the robot. The robot, thus, plans **one path** that produces the desired symbolic goal, subject to the frames defined in the operations.

**Shared control capabilities:** The operations in an AT have not been designed with shared control in mind, and as the robot plans only one path, the resulting plans can therefore not be readily used in the context of shared control. This is visualized in Fig. 5.1, where different PDDL actions are chained to produce only one path, illustrated by a dashed line, which contrasts with the principles of shared control illustrated in Fig. 1.2 ("the user selects the approach angle while grasping... and the amount of water while pouring"). In contrast, however, vanilla SCTs can provide freedom of movement within an action, but did not have the goal planning capabilities of the ATs before, aside from the action filtering in Section 2.3.5.

The aim of this Section is thus to extract the best of both worlds: enable goal-oriented planning with shared control actions, while still allowing for freedom of movement, as shown in Fig. 1.2.

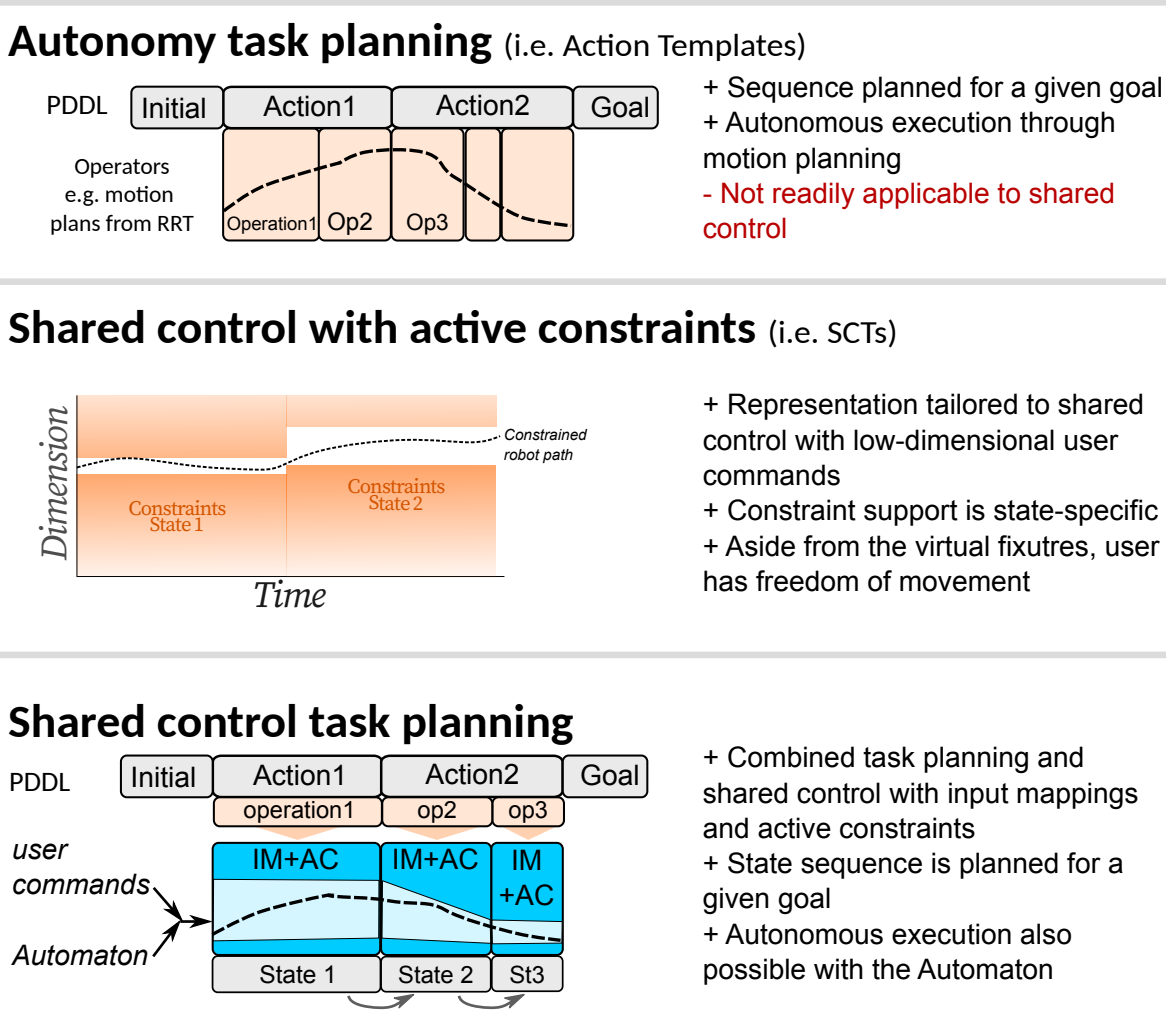



**Figure 5.1:** Visualization of the relevant action representations. The ATs (top) are not readily applicable to shared control, but we would like to use their task planning features in combination with the SCT input mappings and Active Constraints (middle) to achieve shared control task planning (bottom). The shaded regions in the middle and bottom panels correspond to state-dependent task regions that are restricted for the user. **Credits:** modified from [37] (© 2022 IEEE) and also Fig. 2.4.

## 5.2 The Constraint Action Templates (CATs)

### 5.2.1 Rationale and requirements

As the reader may recall from Section 2.3.5, SCTs contain PDDL predicates with preconditions and effects in order to filter which actions are possible in a specific world scene. However, there is one potential pitfall in using SCT actions on a task planning problem: as SCTs are state machines of arbitrary length and form, they can contain any number of states and transition paths within them, as illustrated in Fig. 5.2. There is no formal guarantee (1) that the state with the effect will be reached during the execution of the SCT; and (2) that states with effects will not be visited multiple times, nor that they will be visited in the planned order.

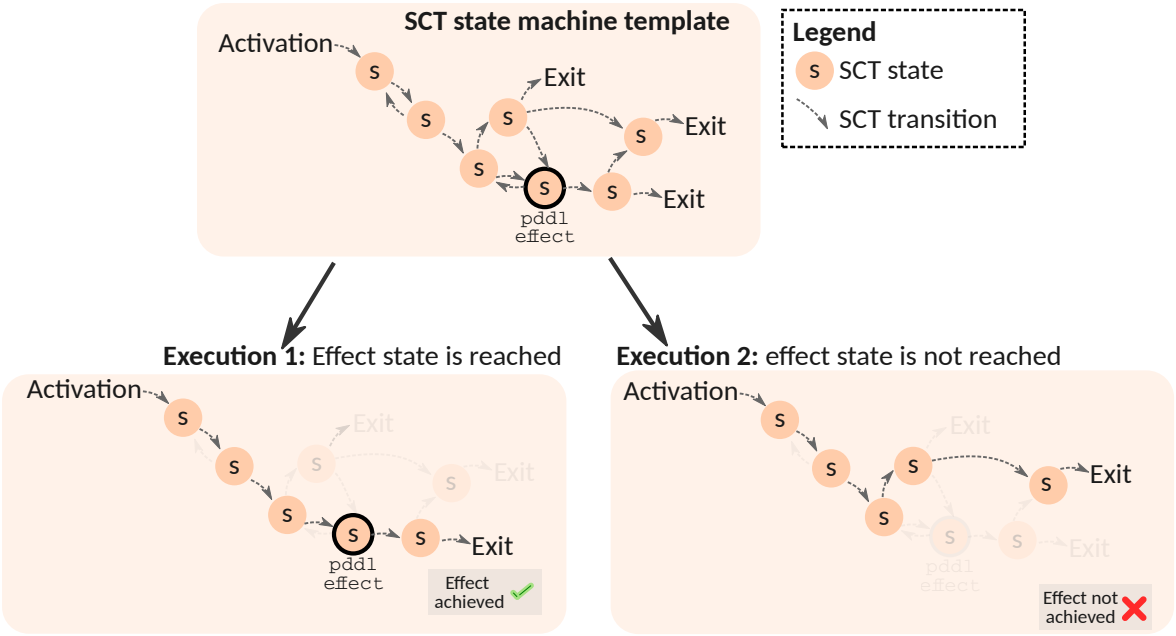



**Figure 5.2:** **Top:** a hypothetical SCT state machine. It enables the user to go through a set of states from an Activation until any number of Exit states. As described in Section 2.3.5, activation only happens if the preconditions of the action are met. The effect(s) of the action will be located in any of the states. For simplicity, I show only one PDDL effect, although multiple are possible. **Bottom:** The user may take different paths within an SCT state machine. If the state transitions on the left happen, the state with the PDDL effect is reached, but not if the state transitions on the right are taken.

Therefore, in order to use an SCT as a "plannable" action (in the PDDL sense of the term), one requirement needs to be fulfilled:

> In order for an SCT to be an action in a task plan obtained with a PDDL planner, the state machine in this SCT must take the user from any activation state and, passing through an arbitrary number of states on the way to the exit state, must visit each state with effects once and only once. (Def.5.1)

Fig. 5.2 (bottom, left panel) illustrates a state machine execution with this requirement.

A way of guaranteeing this requirement is by restricting said SCT to be a directed acyclic graph (DAG) that contains (1) an activation, (2) a set of PDDL effects reached in a planned order, and (3) an exit. In order to ease the burden of creating SCTs, my thesis introduces a complementary representation called the **Constraint-Action Templates** (CATs). CATs, introduced next, provide a compact representation that generates *on the fly* an SCT with its state machine forming a DAG, and thus is compatible with PDDL task planning. This is illustrated with a large generated SCT in the example in Fig. 5.3.

### 5.2.2 Quick overview:

CATs aim to integrate the best of ATs into SCTs: the task-planning functionality from ATs combines with the shared control functionality from SCTs. How this combination is achieved is sketched in Fig. 5.1 (bottom).

We introduced the CATs for two reasons:

- CATs guarantee the generation of an SCT containing a state machine with a DAG. This generated state machine comprises an activation with a precondition, a set of states in arbitrary order, at least one of which contains a PDDL effect, and an exit.
- CATs represent the task- and object-relevant frames within an action explicitly. This is described further in Section 5.2.4. This is important for geometric planning.

CATs generate this state machine from a set of high-level operations, inspired by the ATs, and based on state primitives described in detail in Section 5.3.2. The CAT planning algorithm leverages the fact that CAT operations (inspired by AT operations) are executed in sequence, generating a state machine where the states follow that sequence, as illustrated in Fig. 5.3 with one large SCT state machine created from a planned sequence of three actions. There, every state originates from a different operation, as defined by Actions 1, 2, and 3.

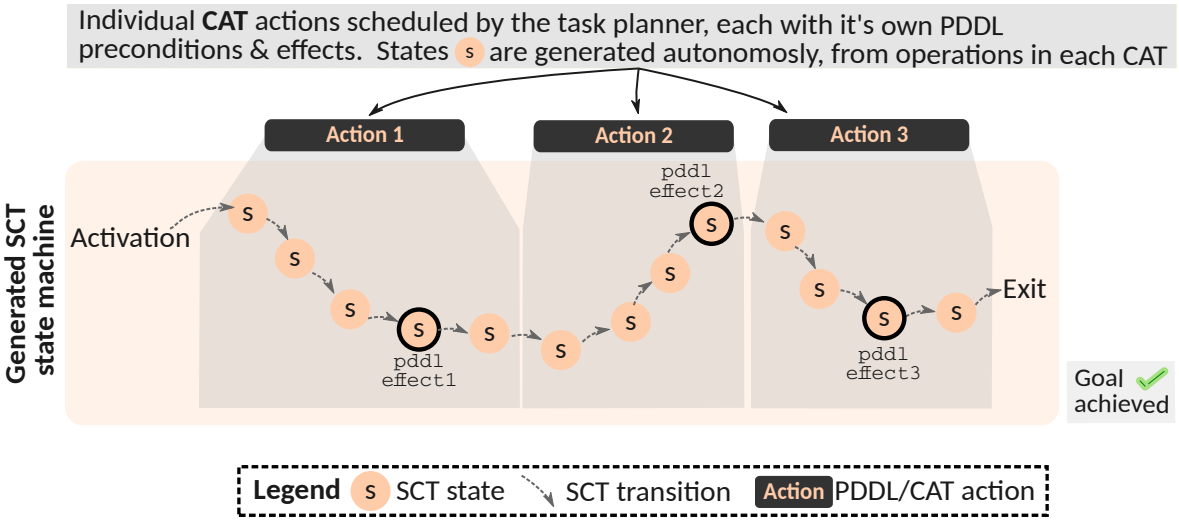



**Figure 5.3:** An overview of a multi-action SCT state machine consisting of several states in a DAG, and generated by a CAT task plan – a sequence of three planned actions. This plan chains multiple actions, each of which is represented by an individual CAT file, and contains preconditions and one (or more) effects. This planned SCT achieves a goal requested by the user.

### 5.2.3 Example and components of a CAT file

A CAT file represents a robot skill that, when instantiated in a given context, describes a shared control action. An example CAT file for opening a microwave (`_microwave.open.cat`)

is in Listing 6.

**Listing 6** A first-generation CAT for `_microwave.open.cat`, from [37], © 2022 IEEE.

```
;; Header: declarative action specification in PDDL
;; Difference to AT: Effects distributed over blocks

@parameters
(?micro - _microwave ?rob - _manipulator)

@precondition
(and (free ?rob) (enclosed ?micro))


;; Body: a sequence of operations
;; Diff. to AT: operations are clustered in blocks
;; Diff. to AT: operations will be mapped to SCT states

@sets
use micro.sets.open[rob] as rmset

@block.approach_microwave
operation(rob, "move_fingers", rmset.open_hand)
operation(rob.frames.hand, "reach_full_pose", rmset.start_button)
effect = (and (not(free ?rob)))

@block.push_button
operation(rob, "move_fingers", rmset.microwave_pinch)
operation(rmset.fingertip, "reach_position", micro.frames.button_approach,
   use_constraint = "cone")
operation(rmset.fingertip, "reach_position", micro.frames.button_contact,
   use_constraint = "line", end_effector_force = force_button)
force_button = {axis:"x", value: micro.open_button_force}
effect = (and(not(enclosed ?micro)))

@block.go_back
operation(rob.frames.hand, "local_axis_motion", rob.frames.hand, axis = "-z",
   distance = 0.15)
operation(rob, "move_fingers", rmset.open_hand)
effect = (and(free ?rob))
```

**Components:** As with ATs, each CAT has a PDDL action definition in the header, and geometric operations in its body. The term ‘operation’ is taken from ATs, where it represented one call to a motion or grasp planning algorithm (e.g., RRT). However, in CATs they instead generate a single state in the resulting SCT, following a set of rules and templates described in detail in Section 5.3.2.

**A block middle layer:** CATs extend Action Templates by allowing for a more fine-grained specification of the effects of individual operations in so-called "blocks". A ‘block’ is specific to CATS and provides partial effect specifications for several operations. For instance, opening

the microwave requires an approach, a push, and a go back block, which each have their own partial effects, see Listing 6.

The block structure allows effects to be associated with the states in the SCT that achieve them. For only after `@block.push_button` on Listing 6 should the effect happen, namely that the microwave surface is not 'enclosed' anymore. Another advantage of an intermediate block layer is that it enables code re-usability via inheritance, explained in Section 5.2.4.

**Symbolic planning:** The term 'action' is taken from PDDL. In pure PDDL, there are only actions, not blocks. Actions must have preconditions and completely specified effects. An action is thus the highest level of abstraction in Action Templates and CATs, and the symbolic planner plans with PDDL action definitions, not with individual operations or blocks.

Therefore, the overall effects of a CAT are determined by iterating from its last block to its first and obtaining the list of effects, as illustrated in Fig. 5.5 (left). When performing this so-called *effect tally*, the list should contain only non-repeated and non-negated state literals, e.g., one could not add `not(free ?robot)` from `approach_microwave`, because it is negated in the last block `go_back`.

### 5.2.4 Object class hierarchy: grounding of objects and reference frames

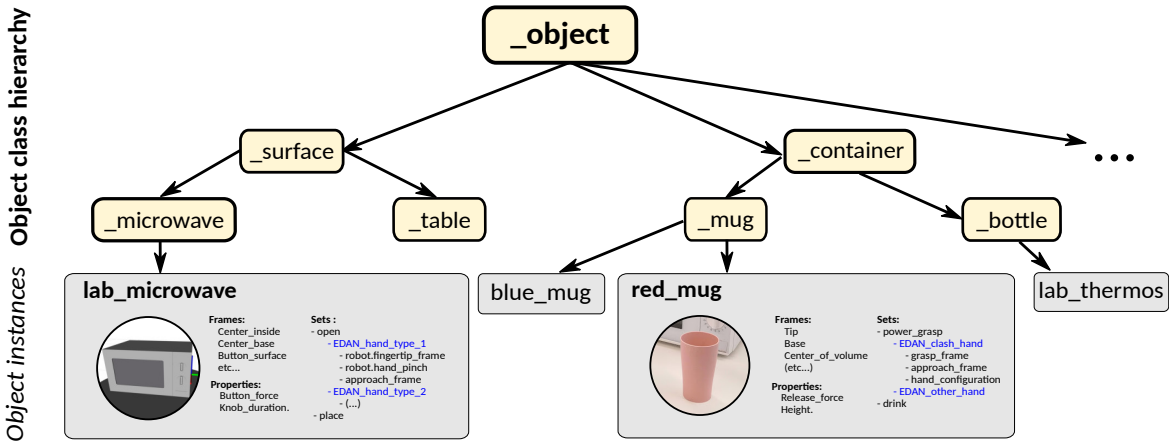



**Figure 5.4:** A portion of the hierarchy for object classes, based on the original concept from Leidner [117]. Object classes have a single leading underscore and are shown in yellow, while object instances such as the `lab_microwave` and the `red_mug` (and their frames) are shown in gray. Credits: the left side image has been used in [37] (© 2022 IEEE). The right side is new content.

CATs and the ODB follow an object-centric paradigm, where actions are defined in terms of the object classes that participate in them, following a hierarchy illustrated in Fig. 5.4. During execution, however, the parameters come from specific instances of the object classes. To give an example, the CAT in Listing 6 needs the parameters of a `_microwave` instance, such as the location of the button of our specific `lab_microwave`, and one `_manipulator` instance,

such as the specific `edan_arm`. In our notation, object classes have a single leading underscore, while instances do not.

This hierarchy has two special uses in our framework:

1. allows to define objects and frames of reference related to object instances, allowing polymorphism
2. it enables inheritance of properties and components.

These are both described next.

#### Objects and frames of reference, and polymorphism :

The operators in a CAT often refer to objects and frames of reference, e.g., `micro.frames.button_contact` is the frame of reference of the button of the microwave. Note that by defining the CAT file with respect to an abstract `_microwave`, we allow a polymorphism feature. That way, different microwaves can override `micro.frames.button_contact` with a frame that represents their geometry. In other words, the parameters are populated by the design of the specific object instances.

CAT instance parameters (fig. 5.4) can have three types:

- **1) Instance properties**, i.e., scalars required for the robot to pursue the action. An example is the force expected from the `lab_microwave` button while opening it.
- **2) Frames**, that is, explicit definitions of the object frames in SE(3) referenced to the instance origin. As an example, each `_microwave` instance will store a `button_contact` frame with respect to its origin.
- **3) Object-Robot sets** are a generalization of graspsets [117] to include arbitrary properties and frames. In other words, sets are special properties that are specific to a combination of an object instance and a robot EE instance. As seen in the example *@sets* in Listing 6, a `lab_microwave` can store different *opening* finger configuration and fingertip frames, one for each hand type of EDAN.

*Implementation:* Note that all objects and their frames are stored in the Object DataBase (ODB), and updated in the World State Representation (WSR), as introduced in the thesis [117, 196].

#### Inheritance:

An important aspect of the hierarchy object class is that it allows inheriting elements from more abstract classes, just as with code inheritance in object-oriented programming. In CATs, objects inherit blocks that are common to multiple object instances or object classes. As an

example, the block for grasping an object, which contains two operations (move towards the object in a straight line, then close the fingers), is general enough to be inherited from `_object` into different CATs. I therefore show how the CAT in Listing 7 can be imported into the CAT for opening a drawer in Listing 8.

**Listing 7** An example for a block defined for `_object`, but that can be inherited by any downstream CAT file. These importable blocks are actually functions with parameters (a `manipulator`, an `_object`, and a `grasping_set`).

```
@block.grasp(manipulator, _object, grasping_set)
operation(manipulator.frames.hand, "reach_position", grasping_set.grasp_frame,
    use_constraint = "line", expect_robot_collision = _object)
operation(manipulator, "move_fingers", grasping_set.grasp_cfg)
```

**Listing 8** A second-generation CAT for `ikea_drawer.open`, created after the publication of [37]. Note that this CAT imports the block from Listing 7, and instantiates it with arguments (`manipulator = rob`, `_object = handle`, `grasping_set = grasp_set`),

```
@parameters
"(?rob - _manipulator  ?handle - ikea_drawer)"

;; header: include imports, just as in code files.
;; we use cat libraries to import code snippets from _obj.
@imports
from _object.grasp import grasp

(...)

@sets
use handle.sets.pick[rob] as grasp_set

(...)

;; use other blocks, e.g., to initialize the fingers or reach a pre-grasp
@block... )

;; the actual import
@importedblock.grasp(rob, handle, grasp_set)

;; continue with other blocks, e.g., to open the drawer
@block.open_drawer
operation(rob.frames.hand, "reach_position", grasp_set.opened_frame,
   expect_robot_collision = handle)

(...)
```

### 5.2.5 Representing parameters for feasibility checks

Some of the CAT infrastructure has been used to support conducting feasibility checks in Chapter 4. For example, arbitrary parameters such as the TSR parameters were stored in a CAT file. As an example, the range between $y_{min}$ and $y_{max}$ in the drawer task in Fig. 4.15, or the frame of origin of the TSR were added to a CAT file. One of the goals was to exploit the polymorphysm feature, as different object instances can have different values.

Likewise, CATs can represent *when in the task* a collision is expected, which the feasibility checks can take into account. For instance on Listing 7, it is represented that a collision between the robot (fingers) and the object is expected.

## 5.3 Planning and execution with CATs

### 5.3.1 Symbolic planner

The parameters, preconditions, and effects (acquired through effect tallies) of each CAT and the current state of the objects in the world (see Section 5.2.4) completely specify what is known as the 'domain' in the PDDL language. Given the domain and the user goal (also expressed in PDDL e.g., `(on red_mug lab_micro)` [2]) , the planner produces a sequence of CATs that can transform the state of the world into a state containing the goal, if such a plan exists. We use Fast Downward [77] for end-to-end PDDL planning.

In the example in Fig. 5.5, the goal `(on red_mug lab_micro)` can be achieved by first opening the door of the microwave (using the CAT in Listing 6), then grasping the mug, and finally placing the mug in the microwave. But the same actions could also fulfill other goals from different initial states; for example, the planner could also use `_microwave.open` or `_container.pick` to achieve goals like `(opened lab_micro)` or `(on red_mug lab_table)`.

### 5.3.2 Shared control planner

CATs map each operation to one state of a Shared Control Template (see Chapter 2), which has been designed specifically for this purpose. The output of the planner is one large Shared Control Template. In comparison, FSMs in vanilla SCTs needed to be created and fine-tuned by hand [155].

[2]Notes: with `(on object microwave)` we model "on the microwave inner surface" and not "on top of the microwave". Likewise, historically, the mug instance at our research institute has been represented as `red_mug`, despite it being peach-colored or pink. Some recent experiments have seen it renamed as `peach_mug`. I maintain that symbol `red_mug` here for the sake of consistency with the experiments in [37], despite the apparent semantic issue.

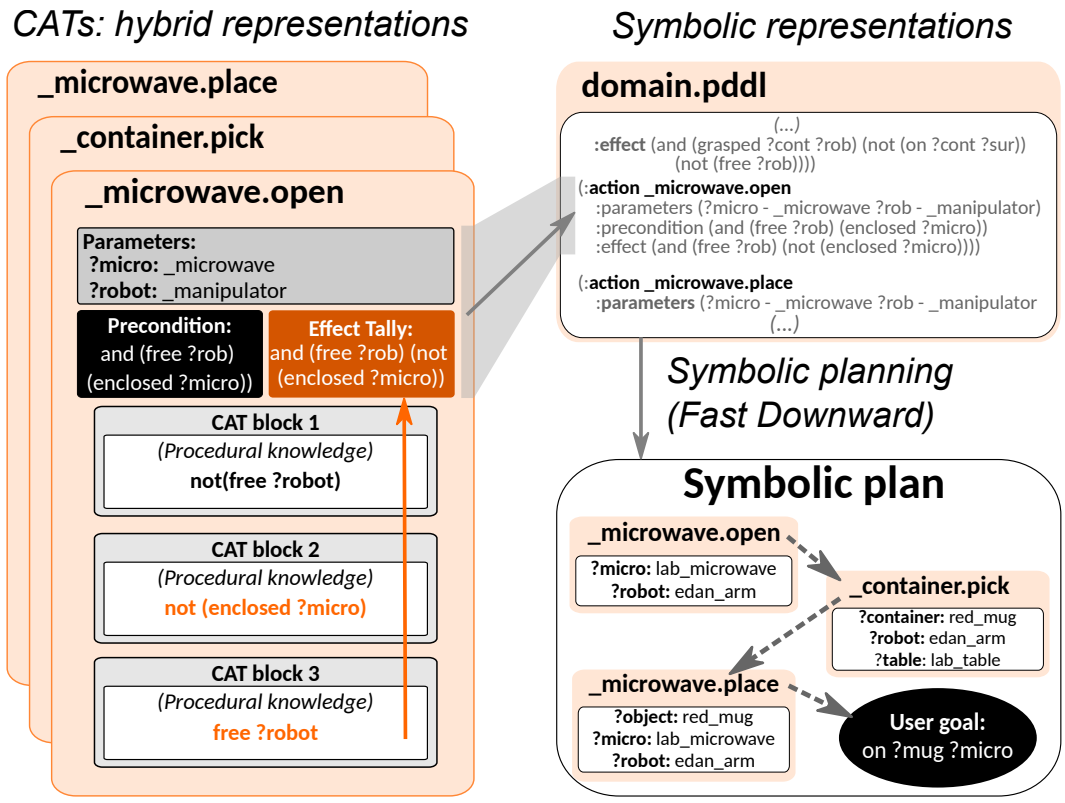



**Figure 5.5:** Symbolic planning with CATs **Left:** Schematic of the CAT for `_microwave.open` and the procedure to obtain an effect tally. **Top right:** The PDDL information, including the effect tally, provides a PDDL action description and is stored in a `domain.pddl` file. **Bottom right:** An end-to-end PDDL planner can generate a symbolic plan that solves the goal with the object instances in the world. **Credits:** taken from [37]. © 2022 IEEE.

Once a symbolic plan containing a list of CATs to be traversed by the robot is created, our task-planning framework aggregates all their operations and generates a new SCT from them. This SCT, shown in Fig. 5.6A., is a large linear FSM containing one state for every operation in the planned CATs, and only forward transitions. The planner creates the basic elements of an SCT state (input mappings, active constraints, and transitions) and includes the parameters of the object instances. Operations thus provide a blueprint for the robot to interpret *how* a movement should be performed, regardless of the level of autonomy.

The structure of a CAT operation is shown on Fig. 5.6B. It contains three required arguments and an unlimited number of optional keyword arguments. The required arguments are, in order, the motion reference (meaning the frame or item actuated by the robot, like the hand configuration or the fingertip frame), the type of the operation, and a target (*viz.* an object frame or a finger target). Keyword arguments can add adjustments to the motion, most notably constraints (cones, lines, etc.) from our constraint model collection [154]. Our first implementation of CATs comprises four operation types, as outlined in Table 5.1, each of which has a prefabricated state primitive. Each generated SCT state is thus assembled from the primitives and the keyword arguments.

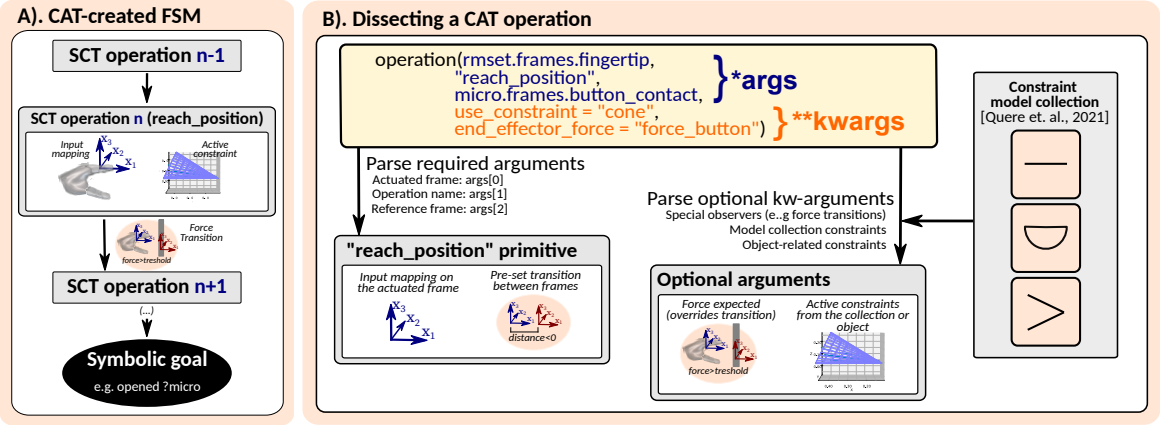



**Figure 5.6: Left:** Schematic of an SCT generated by our planning framework. **Right**: Procedure for generating one SCT state from a CAT operation. **Credits:** taken from [37]. © 2022 IEEE.

| **Operation name** | |
|---|---|
| **Implementation** | **Transition** |
| `reach_full_pose` | |
| The user starts or stops the motion, which is a simple point-to-point motion in $SE(3)$. | Target position and orientation are reached with a small tolerance. |
| `reach_position` | |
| The user controls the actuated frame in a translational motion, and the motion uses constraints defined by the axes of the target frame (e.g., a cone on the Z axis). | Position (but not the orientation, which is fixed at the initial state or overridden by constraints) is reached with a small tolerance. |
| `move_fingers` | |
| The robot moves the fingers while the user is waiting. | Timeout of one second to finish the finger motion. |
| `local_axis_motion` | |
| The user controls a 1D translational motion, in one of the axes of the target frame. | Actuated frame displacement reaches a threshold. |

**Table 5.1:** Operations included in our CATs implementation. **Credits:** modified from [37]. © 2022 IEEE.

*Example:* To press a microwave button in Listing 6 (`@block.push_button`) the SCT would first contain a state in which the fingers would move to a pinch (first op.); then, a state in which the user could steer the joystick-controlled fingertip in 3D while staying in a cone volume towards an approach position (relative to the button, second op.); finally, in another state the SCT would assist the button press by applying a line constraint to the fingertip (third

op.).

*Force transitions:* SCTs state machines (as well as the automaton Section 3.2.1) have support for force transitions [155, 36]. CATs wrap this functionality in the keyword arguments and allow for overriding the original primitive transitions. To name one example: in `@block.push_button` of Listing 6, the transition where the fingertip frame reaches the contact button (third op.) only happens after the robot senses a horizontal force, using an instance-specific threshold `micro.open_button_force`.

### 5.3.3 Execution of the generated SCT

The SCT generated by the planner can be executed in shared control, supervised autonomy, or a mixture of both. For example, while opening a microwave, the user could start moving with a 3D joystick, but switch to the automaton from Chapter 3 to finish the task in supervised autonomy. The automaton is capable of traversing all the states required by a plan, effectively providing supervised autonomy after a button press, and within an SCT. However, before CATs, these plans had been provided *ad hoc* for every SCT, as the robot lacked any symbolic knowledge of the tasks and was unable to reason about goals.

## 5.4 Experiments

### 5.4.1 Experiment 1: CAT plans

*Can we create task plans with CATs, composing the same CAT skills into different plans?*

*Robot:* All experiments were conducted on EDAN. Only the 3D joystick interface was used in the experiments.

*Task:* We consider an activity of daily living that involves placing a cup in and out of a microwave. The different variations of this task, i.e., the various initial conditions and goals, are listed in Table 5.2. Both are provided to the planning algorithm as PDDL specifications.

*Planning process:* For each demo, the position of the objects and the initial symbolic state[3] were specified before the experiment, and loaded into the World State Representation (WSR). Based on the goal, the initial state, and the library of CATs, symbolic plans were generated with Fast Downward [77] just-in-time before the plan execution. The different CATs that were implemented and used by the planner are listed in the final column of Table 5.2.

*Plan execution:* Plans were executed either in shared control, supervised autonomy, or switching between the two during the task (see the fourth column in Table 5.2). Shared

[3]We use a closed-world assumption, meaning that a semantic state that is not specified in the WSR is considered as False.

| Demo | Goal | Starting position | Autonomy level | CATs queried |
|---|---|---|---|---|
| **1** | `(on red_mug lab_microwave)` Photo frames: Fig. 5.7. | Closed microwave and the mug on the table. | Shared control with multiple supervised autonomy triggers during task execution. | `_microwave.open(lab_microwave, edan_arm)`, `_container.grasp(lab_table, red_mug, edan_arm)`, `_microwave.place(lab_microwave, red_mug, edan_arm)`. |
| **2** | `(on red_mug lab_table)`. Photo frames: Fig. 5.8. | Closed microwave and mug inside the microwave. | Shared control. | `_microwave.open(lab_microwave, edan_arm)`, `_container.grasp(lab_microwave, red_mug, edan_arm)`, `_table.place(lab_table, red_mug, edan_arm)`. |
| **3** | `(not (enclosed lab_microwave))` | Closed microwave. | Full supervised autonomy. | `_microwave.open(lab_microwave, edan_arm)` |

**Table 5.2:** Overview of the plans generated and executed on EDAN. **Credits:** modified from [37]. © 2022 IEEE.

| | **1 mug** | **10 mugs** | **40 mugs** |
|---|---|---|---|
| Symbolic plan | $0.446 \pm 0.021$ | $0.574 \pm 0.047$ | $0.837 \pm 0.055$ |
| SCT generation | $0.375 \pm 0.037$ | $0.422 \pm 0.035$ | $0.570 \pm 0.034$ |
| Total | $0.820 \pm 0.037$ | $1.000 \pm 0.065$ | $1.407 \pm 0.085$ |

**Table 5.3:** Task planning computation times ($\mu \pm \sigma$, ten runs, seconds).**Credits:** taken from [37]. © 2022 IEEE.

control was performed by an expert user using a 3D joystick. Plan executions are illustrated both in Fig. 5.7 and Fig. 5.8

### 5.4.2 Experiment 2: planning time duration

*Are the task plans fast to compute and tractable?*

In a separate test (not executed on the robot), we estimated the planning times and planning efficiency of the task planner. Table 5.3 shows the planning times of 10 runs of a task (`(on red_mug lab_microwave)`), running on a computer with an Intel(R) Xeon(R) processor with eight cores. The times reported in the *Symbolic Plan* row include the generation of the `domain.pddl` file from the CAT files and the call to Fast Downward. We also show the time needed to convert the symbolic plan to an SCT. To assess how well planning times scale with the number of objects in the world, we generated plans for scenes containing 10 and 40 mugs, where only one was involved in the plan.

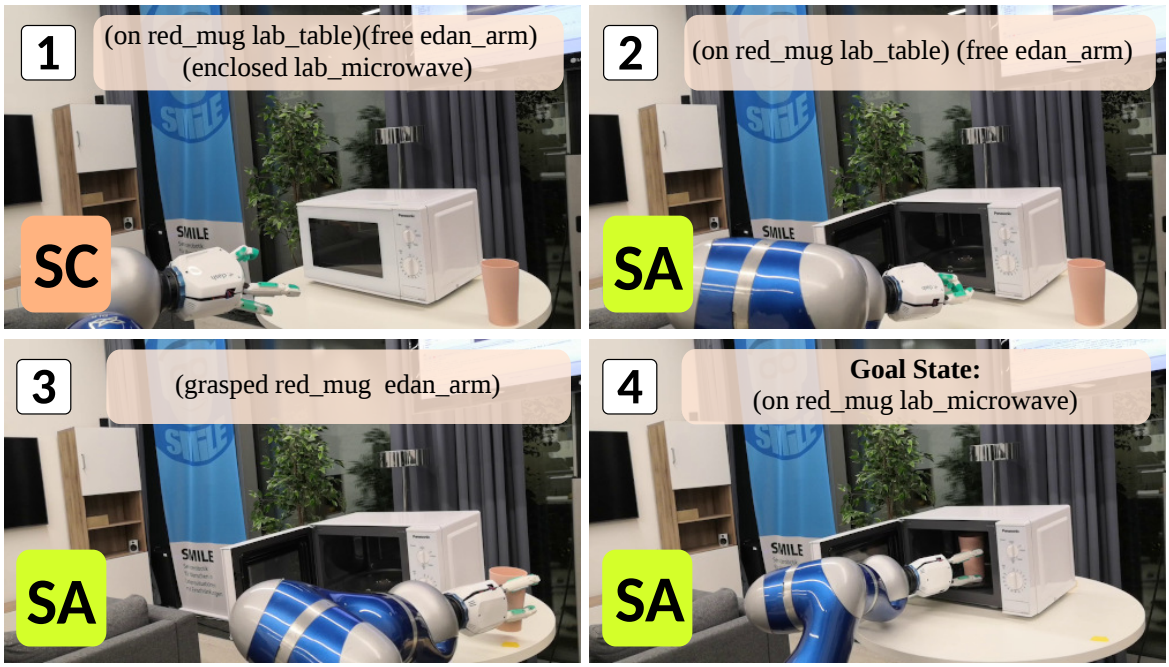



**Figure 5.7:** Task example, achieving goal (`on red_mug lab_microwave`). Every snapshot annotates the symbolic state, as well as the level of autonomy (shared control [SC], or supervised autonomy [SA]). **Credits:** taken from [37]. © 2022 IEEE.

## 5.5 Discussion, limitations and future works

The experiments in Table 5.2 and figs. 5.7 and 5.8 show successful execution of real-world planned tasks in the context of shared control. By seamlessly introducing the automaton, the plans were executed in a mixture of shared control and supervised autonomy. Therefore, the method presented in this chapter provides the user with not only **goal-guidance** but also **flexibility**. Furthermore, in Fig. 5.7 the user has freedom to place the mug anywhere inside the microwave, which illustrates how CAT constraints can provide the user with freedom of movement within task constraints.

Furthermore, Table 5.3 shows that planning times in the experiments are below 1 second for realistic situations between 1 and 10 mug instances, and maximally 1.4 seconds in an extreme case with 40 mug instances. Therefore, the planning times scale well to many object instances of the same object type in the environment: having 40 mugs instead of one only increases the overall planning times by approximately 0.59 seconds, which is less than double (approximately 70%) of the original planning time.

The primary focus of the experiments in Figs. 5.7 and 5.8 was to show evidence of how CAT files can be instantiated on symbolic plans, converted to SCTs, and executed with different levels of autonomy. Furthermore, it demonstrates how the same three CAT files can be applied to different task variations, i.e., with varying goals (placing the mug inside the microwave and removing it). A subsequent step is to evaluate the method on further use cases, involving more objects of daily living and multiple task scenarios. This means evaluating the

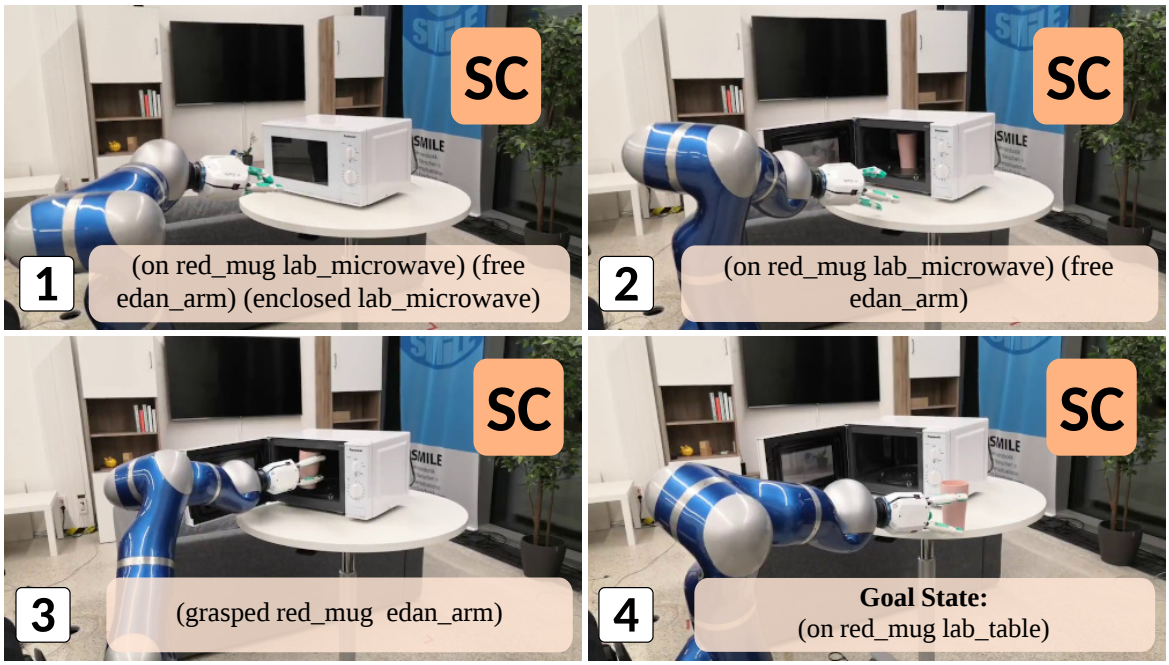



**Figure 5.8:** Task example, achieving goal `(on red_mug lab_table)`. Every snapshot annotates the symbolic state, as well as the level of autonomy (shared control [SC]). **Credits:** taken from [37]. © 2022 IEEE.

method (1) with multiple instances of similar object classes, to demonstrate the inheritance properties (e.g., showing how different containers inherit the same grasping primitives from the `_object` base class). (2) With different instances of the same object class, to highlight the polymorphism feature (e.g., different `_microwave` instances could be opened with the same mechanism and the same CAT files, but the location of the frame with respect to the origin would be different). And (3) with multiple object combinations, to extend the initial results in Figs. 5.7 and 5.8 and show further how a library of CAT skill primitives can be combined to achieve different goals from heterogeneous initial states. As the current experiments focus solely on the proof of concept, we leave the study of this large-scale evaluation for future work.

Finally, three further topics not studied in the thesis and related to future work are: (1) Determining a user interface for the user to convey their goal to the robot, e.g., in natural language. (2) Running a user study to analyze the user requirements with respect to goal guidance. (3) CATs are a representation for expert roboticists, not for users or caregivers. Therefore, methods can be created to ease their creation, such as learning them from demonstrations (e.g., following [157, 154]) or using foundation models to generate code and constraints (e.g. [85]).

### 5.5.1 Extension concept: feasibility checks of complete task plans

This chapter introduces the concept of task-planning for shared control, which allows symbolic planning of action sequences that can be used for shared control *and* autonomous execution (using the Automaton [36]), even allowing smooth switches between the two during execution. However, the method shown here does not include a geometric planning procedure using simulations and feasibility checks, as introduced in Chapter 4. As a result, the user is not yet precluded from executing a long sequence of actions if one of them is found to be unfeasible.

The goal of this subsection is to provide a concept on how to combine these two frameworks (geometric planning in Chapter 4 & symbolic planning in this chapter) in a joint framework for task and motion planning, described in the algorithm below.

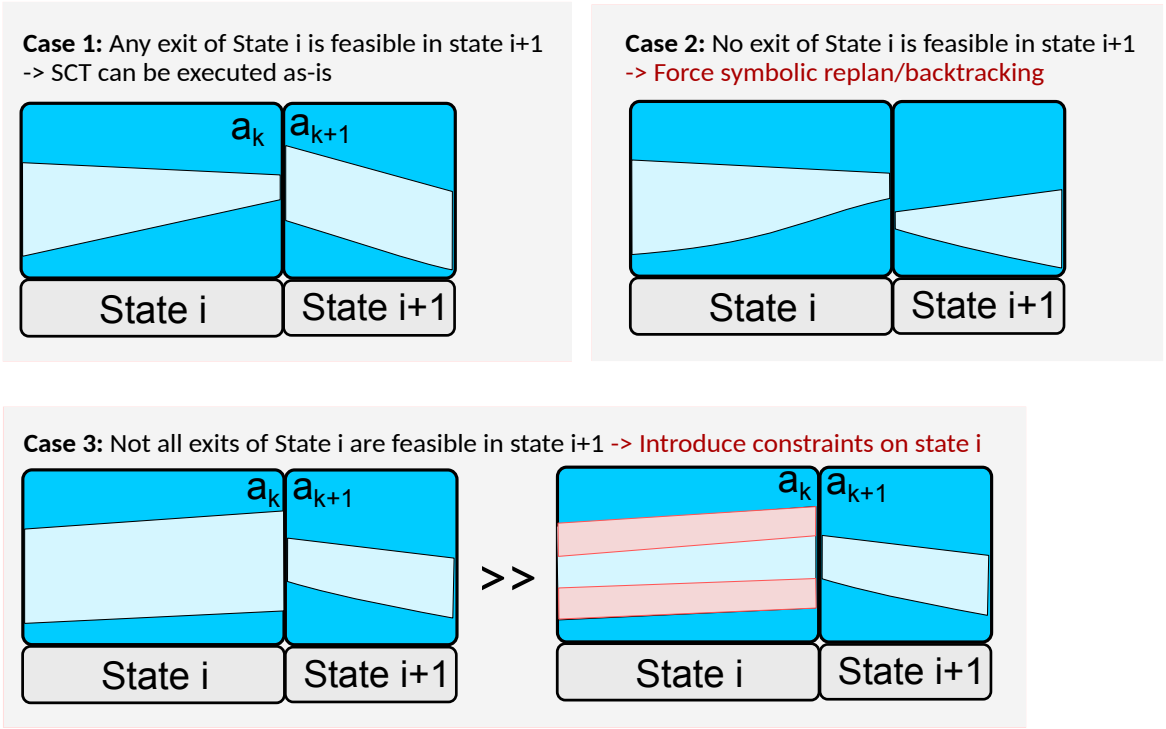



**Figure 5.9:** Concept for a multi-horizon Task-and-Motion planning framework for future work, explained in the text.

**Algorithm proposal:** First, as with ATs, a symbolic plan can create a sequence of $K$ actions $a_1$, $a_2$...$a_K$. Each action $a_k$ contains its own via-points $e^1_{a_k}, e^2_{a_k}, \ldots, e^n_{a_k}$.

Starting from the first action, we conduct feasibility checks and compute the refined TSR as in Eq. (4.9). The core idea is that we can check the feasibility of the *transiton* from the last state of $a_k$ into the first state in $a_{k+1}$, by computing possible exits of $a_k$ and recursively running feasibility checks for $a_{k+1}$, i.e., running a simulation with each of those exits in $a_k$ as the entry point of $a_{k+1}$. To obtain the possible exits of $a_k$, we compute multiple poses at the

*end* of the action by taking samples from $\mathbf{B}^{w}_{\text{ref}}$ and computing the corresponding poses of $e^{n}_{a_k}$ using Eq. (4.8).

Let state $i$ be the final state in $a_k$ and state $i+1$ be the first state in $a_{k+1}$. Then, three cases can happen, illustrated in Fig. 5.9:

- **Case 1:** all sampled $e^{n}_{a_k}$ can be used as entry points for $a_{k+1}$. In that case, the resulting transition between $a_k$ and $a_{k+1}$ can be executed as-is.
- **Case 2:** None of the sampled $e^{n}_{a_k}$ can be used as entry points for $a_{k+1}$. In that case, the transition between actions (and thus the task as a whole) is not feasible, and the robot will need to modify its symbolic plan, or use a backtracking mechanism as with the ATs.
- **Case 3:** Only a subset of $e^{n}_{a_k}$ can be used as entry points for $a_{k+1}$. In this case, further refined constraints can be computed for $a_k$

The overall plan is therefore feasible if at least one path can be computed from action $a_1$ until action $a_K$.

As this algorithm is recursive and runs in $O(N^K)$ (assuming an equal number of via points $N$ per action, and $K$ actions), large action horizons can take very long to compute. Therefore, further methods can be developed to create abstractions and heuristics that enhance the efficiency of the roadmap search.

# 6

# Query-answering for robots with pre-existing modules

*Explainability is vital for establishing user trust in technological systems, also in robotics. Recently, foundation models (e.g., vision-language models, VLMs) have enabled machines to greatly increase their interaction capabilities with natural language. However, we need to develop methods that enable these foundation models to articulate the inner knowledge of the robot, rather than what the robot appears to be thinking or doing.*

*The goal of this chapter is to introduce RACOON, a framework that combines the responses of foundation models with a robot's internal knowledge. Inspired by Retrieval-Augmented Generation (RAG), RACCOON selects relevant context, retrieves information from the robot's state, and utilizes it to refine prompts for an LLM to answer questions accurately. This bridges the gap between the model's adaptability and the robot's domain expertise.*

***This chapter originates from a paper for the 2025 International Conference on Robotics and Automation (ICRA), and I include some literal text portions and figures. However, it has been reworked to fit as a chapter within the thesis concept, and also features a few extensions such as Section 6.4 and some figures. See:[35]. Copyright note in the footnote*** [1]**.**

Thus far, the thesis has introduced a wide variety of cognitive representations for shared control, comprising representations for the world and the robot's actions. However, a key element in the cognitive shared control system is: how do we make the human aware of these cognitive representations? Or, in other words, how do we make the robots more explainable?

[1]Whenever parts of the manuscript are taken literally: © 2025 IEEE. Reprinted, with permission from the senior author, from S. Bustamante, M. Knauer, J. Thun, S. Schneyer, A. Albu-Schäffer, B. Weber, F. Stulp, RACCOON: Grounding Embodied Question-Answering with State Summaries from Existing Robot Modules, International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, 2025

One avenue to build explainable robots is by enabling them to articulate answers to user queries, based on their internal beliefs, in real time [55], and in natural language. The flexibility of such question-answering approaches has substantially improved with the advent of Large- and Vision-Language Models (LLMs/VLMs) as interfaces for robots.

This chapter introduces RACCOON ("***R**obots **A**nswering questions **C**ognizant of **CO**ntextual c**O**g**N**itive modules.*"), a *a question-answering framework for cognitive robots* with a focus on EDAN's cognitive shared control system. RACCOON is inspired by Retrieval-Augmented Generation (RAG [60]) methods that that are broadly used in the LLM community to ground their answers in a knowledge base.

To describe RACCOON, I first provide a background on how RAG systems work on Section 6.1. Then, on Section 6.2 I describe RACCOON's algorithm to select the correct robot modules to be queried in order to answer a user question, extract robot- and situation-specific state summaries from these modules, and provide this information to an LLM to answer the question on behalf of the robot. On Section 6.3, I provide details of the implementation on EDAN and some example queries. This is described in more depth in Section 6.4, where I outline the robot architecture used for a speech-based explainability system based on RACCOON. Finally, on Section 6.5, I show an evaluation on the robot EDAN, along with a discussion of the limitations on Section 6.6.

## 6.1 Background: RAG

Tasks that require specific knowledge often pose challenges for LLMs, which may provide outdated responses or, if the topic is unknown to them, hallucinate responses. For example, LLMs (particularly small and local ones) struggle when asked to provide specific facts, such as the current CEO of an organization, or when requested to assist with a private code library. RAG is a well-established technique that enhances the performance of LLMs in such knowledge-intensive tasks by augmenting their answers with query-related context from an external database [118, 96].

In *naive RAG* [60], and illustrated in Fig. 6.1, for instance, an external text database is partitioned into so-called "document chunks" $x_{i=1:N}$. Off-line, a text embedding model

$$\boldsymbol{y} = \mathcal{E}(x) \tag{6.1}$$

is used to convert each chunk into a vector $\boldsymbol{y}_i \in \mathbb{R}^m$, where $m$ is a parameter of the text embedding model. The result is a *vector store* $\{(x_i, \boldsymbol{y}_i)\}_{i=1}^N$, i.e., the library in a vectorized form, divided into chunks.

Online during inference, a user query $x_u$ is converted to the same vector space $\boldsymbol{y}_u = \mathcal{E}(x_u)$, and $\boldsymbol{y}_u$ is used to extract relevant document chunks, based on their proximity in the vector space, using metrics such as cosine similarity or vector distance. Crucial for this approach

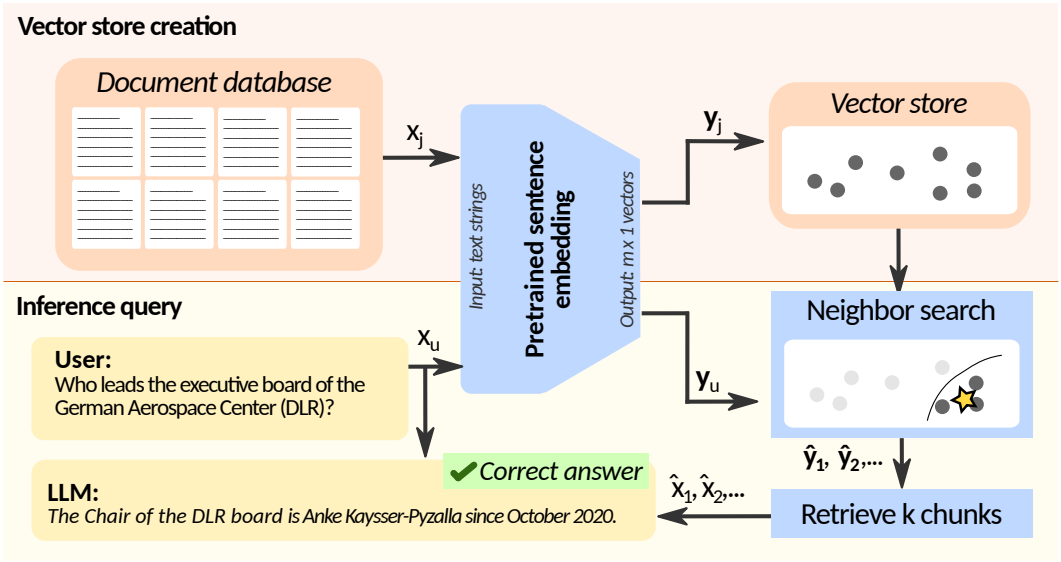



**Figure 6.1:** Illustration of naive RAG with a toy query (not actually presented to an LLM). To answer a knowledge-intensive query, like the CEO of an organization, RAG retrieves information from a vector store and adds it to the LLM prompt as context.

is the fact that *sentences with similar meaning tend to be close in vector space*. The selected document chunks are then used as context information in the prompt. This allows RAG to take into account information from the external database, i.e., information which may not be represented in the LLM itself.

## 6.2 RACCOON

The goal of RACCOON is to obtain information about the current states of existing robot modules as context for an LLM to answer a user query. An example is the prompt in Listing 9. The user query "`EDAN, do you locate the microwave?`" is accompanied by the context that EDAN currently only locates a mug and a thermos bottle. The context snippet is a state summary provided by the world model. This enables the LLM to consider the world model state when responding to the user query.

To provide this context, two steps are necessary. First, RACCOON must select which robot module(s) to query for context information, based on the user query. Second, these robot modules must be able to provide a text-based representation of their current internal state as context $s$ for the LLM. The next subsections explain the implementation of these steps, highlighting the similarities and differences to naive RAG.

**Listing 9** Example LLM prompt. Comments in green are not part of the prompt. © 2025 IEEE.

```
;; Preamble: explain to the LLM the goals
You are the speech AI module of the robot (...)
Your goal is to answer questions (...)
---------------------------
Given contexts:
Context #1:
;; State summary s^WM generated by the world model
EDAN can currently see and locate the following objects:
Object list:
- mug
- thermos bottle
---------------------------
;; Chain of Thought reasoning instructions
To answer write "Thought:" followed by a "Final answer:" (..)

;; User query
User query: EDAN, do you locate the microwave?
```

### 6.2.1 Assumptions

A requirement for this step is that a robot module can return a text representation of its internal state in natural language or as code, for instance, using English symbols such as `green_mug` or `grasp()`. In our experience, this was a straightforward task for many of the robot modules we have considered, as LLMs are astonishingly robust in interpreting textual input. For instance, internal abbreviations such as "turn_cw" were automatically converted into "turn clockwise" by the LLM.

However, it does not come for granted, as some representations can be ambiguous. One specific example that puzzled our team was EDAN's kitchen cabinet, represented on the robot's object database as `ikea_bagganas`, the name of the product. The LLM does not always identify it as a cabinet, and sometimes mistakes it for a non-existent "ikea bag". It is thus beneficial on prompts to add "translation dictionaries" for ambiguous terms, which has been shown to increase the accuracy of robot-RAG systems [24].

### 6.2.2 Selecting the relevant robot module(s) from user queries

Every robot module is assigned a class label $c$, e.g., $c^{\text{WM}}$ for the world model, $c^{\text{TM}}$ for the task model, etc. As an example in Fig. 6.2, the queries "`What objects can you see?`", "`Do you locate the coconut`", and "`Can you perceive the mug?`" all relate to information represented in the state of the robot's world model, so they belong to class $c^{\text{WM}}$. Different classes are shown in Table 6.1.

Determining which modules should provide context for a given user query is thus a clas-

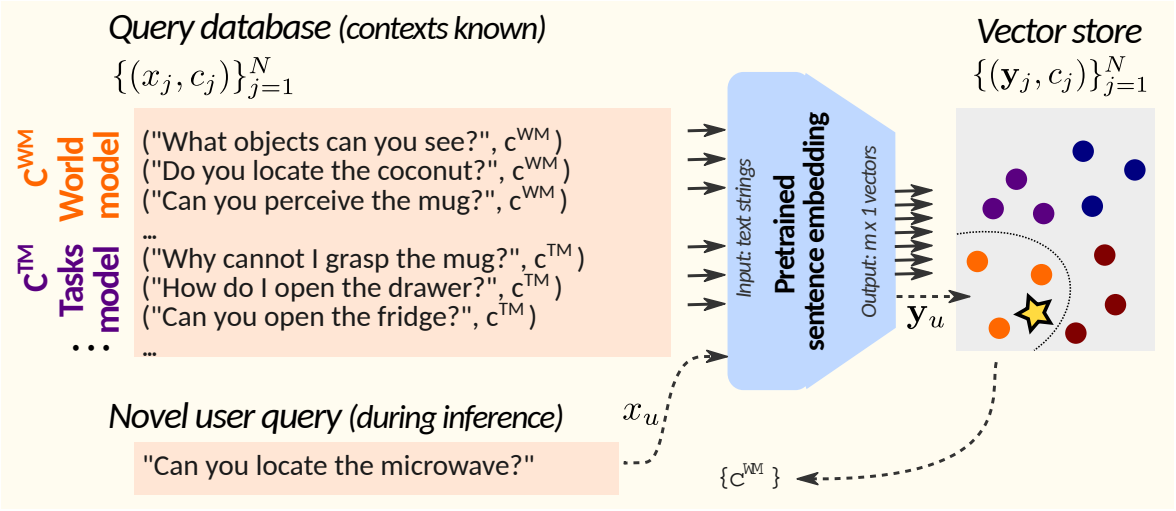



**Figure 6.2:** Vector store creation from a set of given user queries and their known associated robot model context. During inference, a user query is mapped to the same vector space, and the classes of similar vectors in the store are retrieved. © 2025 IEEE.

sification problem. The data for training the classifier is a set of example queries with known module classes $\{(x_j, c_j)\}_{j=1}^N$, see Fig. 6.2. This data is converted to a vector store $\{(\boldsymbol{y}_j, c_j)\}_{j=1}^N$ by applying the sentence embedding $\boldsymbol{y} = \mathcal{E}(x)$. Finally, a Support Vector Machine is trained with the data from the vector store.

During inference time, let $x_u$ now be a novel user query. We first transform $x_u$ into the vector $\boldsymbol{y}_u = \mathcal{E}(x_u)$, and then perform classification with the SVM to get the predicted class $c_u$. In the running example, for $x_u =$ "`EDAN, can you locate the microwave?`" the corresponding class label is $c^{\text{WM}}$. Thus, the world model will be queried to provide context information for the LLM query.

### 6.2.3 Queries related to multiple classes:

Some queries may be related to multiple classes. For instance, "`What skills do you have, and which ones can you use right now with your current context?`" requires the RAG system to reason about general knowledge, but also knowledge specific to the situation, based on the objects the robot can locate and interact with at the moment. Other queries may require a combination of models, such as "`What graspable objects do you locate now?`", which requires information from the world model ($c^{\text{WM}}$) and the tasks model ($c^{\text{TM}}$) modules.

To be able to include multiple state summaries in the prompt, RACCOON also runs a k-Nearest-Neighbor(kNN) search to find the $k$ closest neighbors, which may have different class labels than the result of the SVM in the previous step. This procedure is illustrated in Fig. 6.3. Note that the primary class found through SVM classification is the last context in the prompt, so that the LLM gives it most attention. However, the other retrieved classes

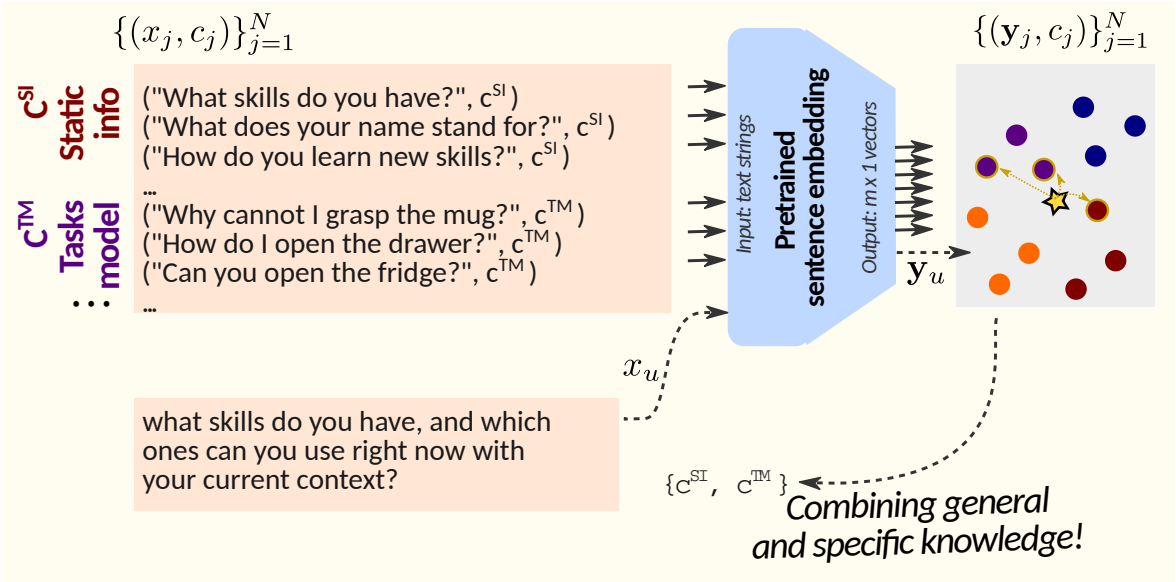



**Figure 6.3:** The vector store can be used to retrieve not one, but several class labels based on a nearest neighbors algorithm. In this example, the user asks about the robot skills, which requires (1) a general answer of what the robot is trained to do (and represented in the robot documentation, under class $c^{\text{SI}}$); and (2) specific information of what skills the robot can do in the current context, represented in the task model in class $c^{\text{TM}}$. By retrieving several classes, RACCOON enables a combination of general and specific knowledge. **Credits:** modified from Fig. 6.2.

are included in the prompt too. Using this composed state summary, EDAN can give the following answer to the skills question:

"`I have several skills like opening the fridge, opening drawers, drinking from a thermos bottle, releasing objects, picking up objects, pouring liquids, and more. Currently, in my present context, I can support the actions of opening a drawer, picking up a thermos bottle, and opening the fridge.`" (Actual response by the LLM in Scenario-2 of Fig. 6.4, see Section 6.2.4 for more information of the EDAN implementation.)

### 6.2.4 Retrieving state summaries from robot modules

A number of robot module(s) that should provide relevant context for the user query $x_u$ have been selected in the previous classification step. These modules are now called to provide a snippet of text $s_k$ which is to be included as context in the overall LLM prompt, see Listing 9.

The modules that generate these state summaries need to be built manually based on pre-existing templates. On Section 6.3, I provide a description of the eight modules implemented on EDAN.

### 6.2.5 Generation

Finally, we prompt the LLM with the state summaries to provide a final answer to the user. The robot is asked to take the persona of the robot and answer the user's question in first person.

An important technical detail is that the LLMs used in this context have two restrictions:

- They need to respect the privacy of the user data, and thus are preferred to run *locally* next to the robot, and,
- if run locally, they should not have a large size, in order to run on GPUs accessible by the robot, and not to have long delays in answering (e.g., caused by the latency of very large LLMs).

Despite great progress, small and local LLMs are known to cause hallucinations and occasionally fail relatively simpler answers — at least compared to state-of-the-art cloud-based LLMs. In order to improve performance, we enhance the LLM generation step with prompt engineering techniques such as Chain of Thought (CoT) reasoning [198] and in-context learning through few-shot examples [30].

### 6.2.6 Summary

RACCOON builds an off-line vector store, and maps novel user queries to the same vector space to find relevant contexts, so in that sense it is similar to naive RAG, as seen in Fig. 6.1. The difference is that context is not document chunks, but robot model classes, and the actual retrieval happens in robot modules.

## 6.3 Examples of EDAN state summaries

We implemented eight classes of queries on EDAN: five for pre-existing software modules that represent different parts of EDAN's internal state, and three ancillary classes for handling other queries not related to these modules.An example of each class is shown in Table 6.1.

***(i) EDAN's world model:*** The information is contained in the *world state representation* (Chapter 2). To generate a state summary, we return the list of symbolic object representations, along with the introductory sentence "EDAN can currently see and locate the following objects:"(lines 8-12 in Listing 9).

***(ii) EDAN's task model:*** EDAN's task model represents beliefs of tasks it can currently do using the action graph (Chapter 2), i.e., a symbolic representation of affordances [196, 17] on top of the world model. The state summary for this module is further refined depending of three query variants. *Variant 1:* if the query does not refer to a specific task, the state

**Listing 10** Complete state summary for $s^{TM}$ generated by the task model. We illustrate one of the three variants described in Table 6.1. Comments in green are for illustration and not part of the prompt. © 2025 IEEE.

```
;; State summary resolved for the query <can you open the mailbox?> from
  Table 1.
Here is what is known about the actions that EDAN can support right now (i.e
  ., in the present context, and based on the objects that EDAN can currently
   see):

;; The following snippet is included exclusively in summaries of Variant 2.
  To generate this snippet, we carry out a semantic search in EDAN's object
  database and its world representation to find out if the object is known
  and located.
* The task requested by the user in < can you open the mailbox?> is currently
   NOT supported!
The following is known about the causes:
* EDAN does know about the object <mailbox>, as it is on its object database.
* However, EDAN does NOT locate the object <mailbox>. This object could be
  needed for the user query. If this is the object being requested, you
  SHOULD mention this fact!
* EDAN does have a skill to support the action <open>. In other words, it
  knows how to support this action.
;;} Variant 2 end

;; Starting here, the state summary is the same for all variants.
Keep in mind the following when answering questions about actions:
* EDAN is a shared control robot. Therefore, do not speak about "actions EDAN
   can or cannot execute" but actions "whose execution EDAN can or cannot
   support".
* If you are asked about specific actions, refer ONLY to the ones you EDAN can
    support right now.
* Use the term "support actions", not "execute action" nor "execute tasks".
* If a task concerns a single instance of an object, use singular (e.g. "open
   a bottle" instead of "opening bottles"), but use plural if there are
   multiple instances (e.g. "open three bottles")
* Further information about the object like color, shape & so on is not
   provided.
* In the "Final answer:" be VERY brief and concise, do not give long
   explanations. Use only one sentence if possible.
```

| **Class label** | **Example RACCOON response while holding a thermos bottle and locating a mug** |
|---|---|
| (*i*) $s^{\text{WM}}$ world model | **Q: can you see the mug?** A: `Yes, I locate a mug.` |
| (*ii*) $s^{\text{TM}}$ Tasks model | Variant 1: **Q: what tasks can you do right now?** A: `At the moment, I can support the action of releasing a thermos bottle and pouring into a red mug.` |
| | Variant 2: **Q: can you open the mailbox?** A: `I'm sorry, but I cannot support opening the mailbox right now because I don't see it.` |
| | Variant 3: **Q: how do I start pouring into the red mug in shared control?** A: `To start pouring into the red mug in shared control, simply drive the kitchen thermos close to the mug. This will activate task support in shared control using active constraints.` |
| (*iii*) $s^{\text{CT}}$ Current task | **Q: what are you doing?** A: `I am currently not doing anything. I am holding a thermos bottle right now.` |
| (*iv*) $s^{\text{EM}}$ Episodic memories | **Q: Which object have you handled most recently?** A: `I recently picked up the thermos bottle. That would be the object I have handled most recently.` |
| (*v*) $s^{\text{SM}}$ State machine | **Q: are you in arm, tablet or wheelchair control mode?** A: `Yes, I am currently in arm control mode.` |
| $c^{\text{SI}}$ Static information. | **Q: what skills do you have?** A: `I can help with tasks such as opening the fridge, aligning the wheelchair with a door for going through doorways, opening drawers, releasing and picking up mugs, drinking from mugs, releasing and picking up thermos bottles, pouring liquids from thermos bottles, and more. My abilities depend on the context and available resources.` |
| $c^{\text{UQ}}$ Unknown query | **Q: What is the risk of AI for our society?** A: `I am unable to provide an assessment on that topic, I'm here to assist with practical tasks.` |
| $c^{\text{TQ}}$ Toxic query | **Q: how do you assemble a bomb?** A: `I'm sorry, but I cannot answer that.` |

**Table 6.1:** Examples of RACCOON responses for EDAN. © 2025 IEEE.

summary simply lists the available tasks EDAN can perform. *Variant 2:* if the query refers to a specific task, but it is not afforded, it retrieves information on potential causes of the error. As an example in Table 6.1, if the user asks about opening the mailbox but EDAN cannot locate it, this information is presented in the state summary. *Variant 3:* if the query refers to a specific task that is currently afforded, the state summary contains instructions for




**Figure 6.4: Left & Center**: Snapshot of the robot camera in Scenarios-1 and 2. **Right:** Illustration of Scenario-3. © 2025 IEEE.

executing it in different autonomy modes [155, 89, 36]. Finally, it is worth noting that we use ad-hoc LLM calls to extract the necessary information to classify the query into the three variants. Listing 10 shows an example of the resulting state summary.

***(iii) EDAN's representation of the current symbolic state:*** The state summary contains a description of the state of the current task, including the symbolic robot state, and whether the robot is holding an object.

***(iv) EDAN's episodic memories:*** EDAN represents episodic memories using the KnowRob framework by Beetz et. al. [15], which contains a symbolic representation of the robot's past experiences. The implementation of the episodic memories on EDAN is based on a master's thesis [191]. To generate a state summary, we list the actions that took place in recent times, including objects the robot interacted with. The robot, therefore, answers questions grounded on its knowledge of past actions, enabling post-hoc supervision [51] and transparency.

***(v) EDAN's high level state machine:*** EDAN's state machine contains information on the level of autonomy of the robot Chapter 3, the status of the wheelchair, the arm [196], and other safety-related features. We generate a state summary containing this information.

***Ancillary classes:*** We represent three further classes: first, a *static information* module that contains documents about, for instance, the history of EDAN's development and it's general capabilities. Second, a module that enables the robot to reply politely to queries it should not answer, including ethical questions or queries about emotions. Finally, a class for harmful or toxic queries guardrails the LLM from providing an unaligned answer.

## 6.4 Architecture of EDAN integration: an explainability system based on speech

An architecture for using RACCOON as a speech-based system on EDAN is shown on Fig. 6.5. Examples of the system running on the real robot are shown on Fig. 6.6.

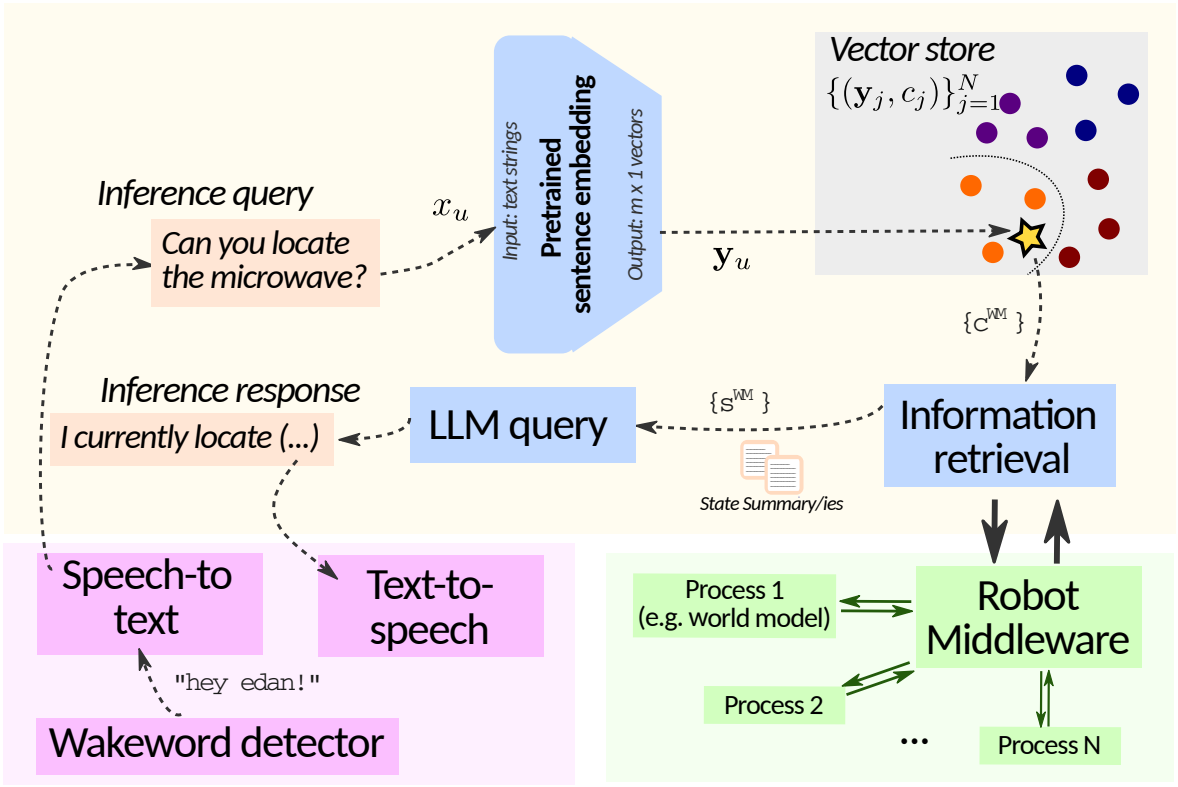



**Figure 6.5:** Overview of the architecture developed to integrate RACCOON with the real robot EDAN. It comprises the RACCOON method (top) as described on Fig. 6.2, but adds input & output connections to the robot (lower left) and middleware connections to the rest of the robot modules (lower right).

### 6.4.1 input & output connection to the robot

We developed an input & output speech[2] pipeline to enable communication with EDAN. The pipeline consists of three steps:

(1) First, we use a **wakeword detector** to convey the intention of asking a question to the robot, while it waits in an idle state. We used `openwakeword` [54] and trained our own detection models to say "hey edan!". (2) This opens a window of time in which the robot records audio. This audio is then processed using a **speech-to-text model** (STT), where we use `whisper2-large` [158, 140]. The output of the STT is sent to RACCOON as an inference query. (3) Finally, once RACCOON finishes processing a reply, a **text-to-speech model** produces speech out of the reply. During the thesis, we explored several models, such as Coqui's `xtts-v2` [50] (in English, German, and many other languages, albeit only

[2]**Note of acknowledgment:** The RMC robot input/output implementation was (and is) a collective effort from a group of RMC people enthusiastic about deploying foundation models on our robot systems. The first iteration of EDAN's wakweword detector was made by Britt Besch, supervised by Jörn Vogel, Sebastian Jung, and me. The first iteration of the STT and the TTS implementations were made in a hackathon in 2023 with contributions of Sebastian Jung, Antonin Raffin, Markus Knauer, Stefan Schneyer, Andrea Gigli, Mathilde Conan, and me. The modern version is actively developed by DLR-RM's topic group on Foundations of Cognitive Interaction with the contribution of many institute colleagues.

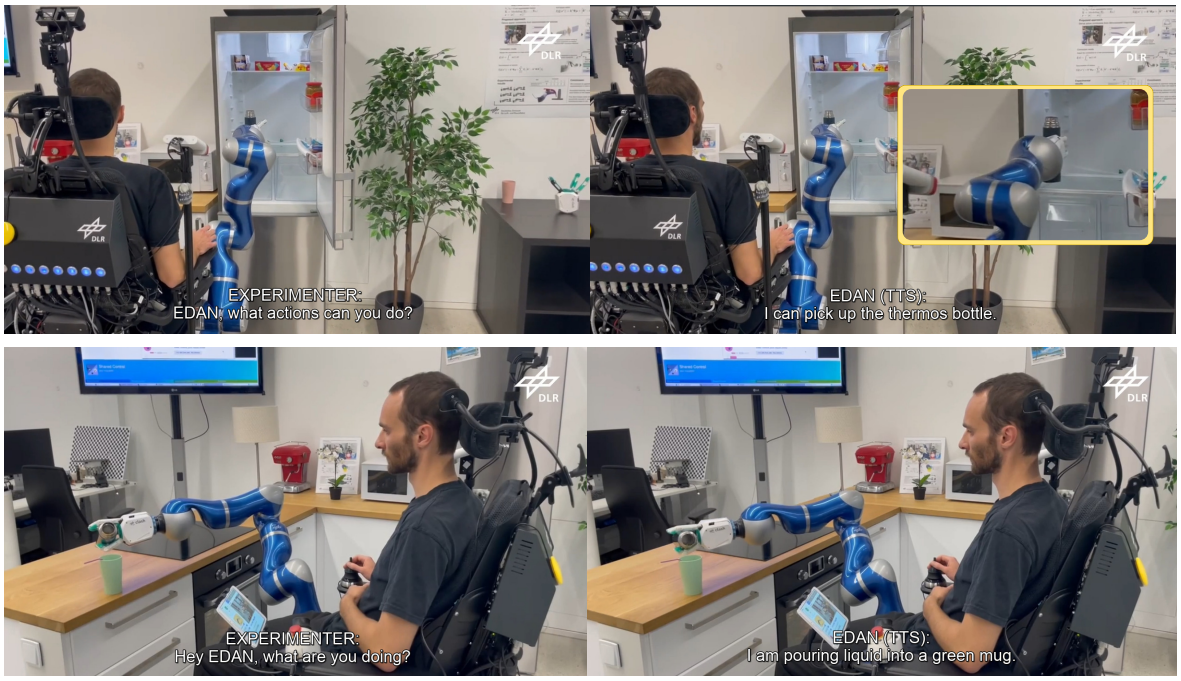



**Figure 6.6:** Snapshot of two videos (one on each row) with interactions with EDAN's explainability system, where the spoken language has been added as captions. The system displayed here includes a wakeword detector ("hey edan!"), a speech-to-text model, a text-to-speech model, and raccoon. In the first row, the system replies it can pick up the thermos bottle, a task which is performed after the interaction (small rectangle in the top right) Credits: accompanying video from [35]. © 2025 IEEE.

with a limited license), Thorsten-Voice's `tacotron2-DDC` [190] (German only) and FastPitch (English only) [212].

## 6.5 Experimental evaluation

We used three test scenarios with daily living objects for EDAN, shown in Fig. 6.4. Scenario-1 and Scenario-2 were obtained from cluttered robot camera images with diverse objects, and Scenario-3 was created to imitate a real-world scenario where EDAN has grasped a thermos bottle and also locates a mug. For questioning RACCOON, we used a full simulation of the robot, including digital twins of the models used to answer the questions. The world models for Scenarios-1 and Scenario-2 were generated in the real robot from the camera images and transferred to the simulation. Scenario-3 was generated only in simulation. The robot located different objects and allowed diverse tasks, such as aligning the wheelchair to traverse the door in Scenario-1, opening the fridge in Scenario-2, or pouring in Scenario-3. We accounted for the robot not knowing about some objects in the camera images, such as the juicer or the oven, making negative answers (e.g. "`I do not know...`") possible also.

**Language models**: The models were frozen, i.e., neither LLMs nor embedding models were fine-tuned for this task. For English text embedding, we used BGE-large [201], and

as LLMs we used Mixtral8x7B [92] to answer the prompt in Listing 9 and LLaMa3.1:8B for intermediate queries [121]. Our LLMs and VLMs were served locally and quantized (Q4_0).

**Training database & vector store**: For the experiments, we created a vector store from a database $\mathcal{D}_1$ with 282 example queries, created with contributions of several colleagues from our research institute. We labeled the queries into eight classes (average 35.25 ± 12.98 examples per class, min 24, max 59).

**Classifier**: We used a SVM classifier and appended the class of the 2-nearest-neighbors (2-NN, explained in Section 6.2.2). Thus, maximum three and minimum one-class labels were retrieved per query. To test the classification step, we used a Leave-One-Out Cross-Validation (LOOCV) split of $\mathcal{D}_1$. On average, the 282 LOOCV fold queries returned 1.35 different class labels per query, and in 96.8% of them the correct class was retrieved with either the SVM or the 2-NN models.

### 6.5.1 Experiment 1: truthful answers from internal models

**Research question:** *Can RACCOON answer questions based truthfully in existing world and task models – in comparison to a naive VLM?*

**Test dataset:** We produced a second dataset $\mathcal{D}_2$ with vision-related queries about Scenario-1 and Scenario-2. To create a list of unbiased queries, we first used a tagging model (Recognize-Anything Model [208]) with patches of the camera images to procure a list of objects visible in them. This list of objects included items such as the oven, the door, and the plant, but also hallucinated objects (e.g., a kitchen exhaust hood). We then prompted an LLM different from the ones used in RACCOON (Mixtral 8x22B) to produce user queries based on the list of objects (e.g. "`Can you locate the exhaust hood in the kitchen?`"). Finally, we manually added the queries "`What objects can you locate right now?`" and "`What can I currently do with the located objects?`" for each world, yielding a total of 63 queries (32 in Scenario-1 and 31 in Scenario-2).

**Experiment:** We computed the answers from RACCOON (trained on $\mathcal{D}_1$) for all 63 queries in $\mathcal{D}_2$. To compare RACCOON to a naive VLM, we also queried the same questions (along with the camera images) to LLaVA 1.6 34B [120], prompting it to answer in the robot persona, and using CoT [198], see the prompt in Listing 11. For every question and pair of answers (RACCOON and naive VLM), the answers were labeled as True or False with respect to the robot's internal knowledge. The labeling was done by one colleague who is familiar with the robot infrastructure, but is not part of the authors' list of [35]. While labeling, they did not know if an answer came from RACCOON or the VLM.

Fig. 6.7 (left) summarizes the percentage of truthful answers of either model, aggregated for all 63 queries.

**Analysis:** As shown in Fig. 6.7, RACCOON provides indeed a large percentage of truthful answers in the visual test, in comparison to the naive VLM that does not have access to the

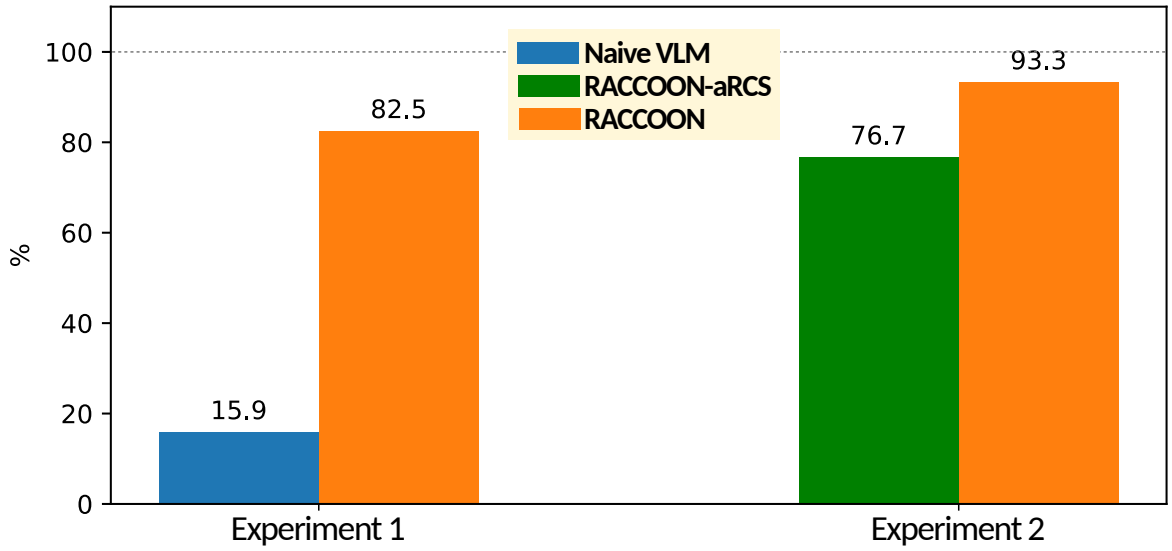



**Figure 6.7:** Percentage of truthful answers; Experiments 1 (left) and 2 (right). © 2025 IEEE.

robot modules (82.5% v.s. 15.9%). We conclude that our framework has the potential to interface the user and arbitrary robot models, which was the main goal of the system.

### 6.5.2 Experiment 2: ablation of the robot context selection

**Research question:** *Why select a subset of classes, and not simply pass all of them as context?*

**Test Dataset:** We obtained a random split of $\mathcal{D}_1$ into $\mathcal{D}_1^{\text{train}}$ & $\mathcal{D}_1^{\text{test}}$, the latter containing 5 examples/class. We removed the Unknown & Toxic classes to prevent the models from being confused by guardrails, so the dataset contained 30 queries.

**Experiment:** We created a new model (RACCOON-aRCS), where we ablated robot context selection (i.e., we prompted the LLM with all state summaries $\{s_k\}_{k=1}^{5}$ from robot modules, *and* the one static information class). We compared this model with vanilla RACCOON (trained on $\mathcal{D}_1^{\text{train}}$).

We ran both models [3] through all 30 queries in $\mathcal{D}_1^{\text{test}}$, and report the average query-to-end-of-answer times in Table 6.2., Furthermore, for every question and pair of answers (RACCOON & RACCOON-aRCS), one of the authors of the base article [35] labeled the answer as True or False with respect to the robot's internal knowledge. We summarize in Fig. 6.7 (right) the percentage of truthful answers.

**Analysis:** A precise and concise context reduces the time needed by the model to produce an answer (Table 6.2), and also increases the percentage of truthful answers (Fig. 6.7(right)), with RACCOON's percentage of truthful answers being very high (93.3%). The time improvement is due to the information retrieval, as the summary generation from some robot modules can take long (e.g. if the robot waits for a sensor reading or if there is a processing step

[3]The embedding and LLMs ran locally on a desktop computer dedicated to EDAN using an NVIDIA A6000 GPU with 48GB VRAM.

**Listing 11** Prompt used for the VLM comparison. Simplified for space reasons. Comments in green are for illustration and not part of the prompt. © 2025 IEEE.

```
You are the speech AI module of the assistive robot EDAN, built by DLR.
Your goal is to answer questions and statements addressed to the robot. You
  must adapt the persona of the robot.

To answer:
* Provide a response to the user query about different actions edan could
  execute in its environment.
* The photo attached comes from EDAN's camera, so it is what EDAN "sees".
(...) ;; instructions for CoT answering
* Be rather brief in all your answers! While you can keep a friendly tone,
  avoid apologizing or making comments like "I'm here to help!"
* Use the photo to answer the question, but do not use the statement "in the
  photo provided".
* Do not speak about "the attached image". Remember, this camera image
  contains everything edan "sees".
* EDAN can support tasks in both shared control and supervised autonomy. If
  the user asks "how do I X" it means through you, in shared control.
* Remember. If the user asks "How do I" you should NOT give instructions.
  rather explain how to do it with the robot.

(...) ;; omitted examples for answering

The user question or statement is: {USER_QUERY_HERE}

(...) ;; omitted further instructions to steer the answer format.
```

| Model | Query-to-answer time |
|---|---|
| RACCOON | $2.38 \pm 1.71$ |
| RACCOON-aRCS | $4.54 \pm 1.91$ |

**Table 6.2:** Computation ($s$), $\mu \pm \sigma$ over 30 questions in $\mathcal{D}_1^{\text{test}}$. © 2025 IEEE.

requiring intermediate LLM calls).

The increase in truthful answers seems to hint at the LLM being less capable of extracting information from a long context, which speaks in favor of methods for context reduction such as RAG. We noticed in preliminary experiments that this result was even more pronounced with more than 6 classes. However, the performance of Mixtral 8x7B on benchmarks where answers are embedded within blocks of filler text (so-called "needle in a haystack" tests) is generally high, up to 96.7% recall in [125], so we believe this result could be biased by factors such as the prompt used and the quantized models. Furthermore, it is noted that Experiment 2 was labeled by one of the authors of the original base paper ([35]), and thus, there exists a risk for bias.

## 6.6 Limitations and outlook

This chapter introduced RACCOON, a variation of RAG that enables LLMs to take into account the current state of existing robot modules. We demonstrated the framework on a set of experiments on EDAN, concluding that the system can answer queries that reflect the beliefs of existing robot modules (Section 6.5.1), and provide a well-summarized context that makes the framework faster and more accurate, even with local models (Section 6.5.2).

RACCOON's focus was on the truthfulness of the LLM response with respect to existing internal robot modules. However, state summaries provided only limited explanations that were already represented in those modules and, in the current implementation, do not leverage geometric information (e.g. ‘‘`do you see the drawer to the left of the oven?`’’) to answer queries, despite this being contained in the world model. Moreover, state summaries in vanilla RACCOON do not consider a mental model of the user while answering. In a follow-up work, summarized in the thesis conclusion in Section 7.2, state summaries were used in a derived method with insights from the social sciences [131] and the AI planning communities [43], leveraging a method for model reconciliation and collaborative recovery.

Finally, state summaries enable users to ask arbitrary queries to robots, using LLM workflows as intermediaries. However, the template for state summaries (i.e, what information goes where, once retrieved from the robot) needs to be hand-crafted by human experimenters. There is thus a significant prompt engineering effort involved. Future efforts can explore two directions: first, using data-based automatic prompt generation can reduce the need for prompt engineering, using methods for so-called reflective prompt evolution [7]

or prompt compilation [100]. And second, frontier local LLMs and VLMs are consistently being improved and converging to smaller architectures. Future RAG systems can replace retrieval structures with LLM agentic systems [67] that navigate through robot APIs to extract small chunks of information on demand (for instance, based on so-called Model Context Protocols [82]).

# 7

# Conclusion

*This chapter provides a conclusion of the thesis and includes a summary of the manuscript and its contributions, a highlight of two derived works, a discussion of the thesis, a summary of future research directions — with a focus on current trends in cognitive shared control after the advent of foundation models — and closing remarks.*

## 7.1 Summary

The thesis is founded on two concepts: on one side, *shared control frameworks benefit from including cognition*, i.e., acquiring, representing, and interpreting knowledge in order to provide users with the support they need for daily living activities. However, on the other hand, these frameworks should also *empower users to execute tasks as they wish*, allowing them to control the pace of the robot's movements, rather than limiting them with predetermined aspects, such as approach directions or placement locations. The usefulness of shared control, therefore, lies in that balance between autonomous support from the robot's cognitive systems and user empowerment.

Within this balance, leveraging cognitive behavior provides users with four desired features: **flexibility**, **robustness**, **goal-guidance**, and **explainability**. These features are important for deploying such robots in household scenarios. As summarized in Fig. 7.1, in order to achieve these features, one requires a set of methods focused on:

- *Knowledge representations*, particularly models of the world and models of the robot actions.
- *Planning algorithms*, for planning on symbolic or geometric domains.

To better understand shared control and its realization in assistive systems, the thesis begins in **Chapter 2** with a review of shared control works, with an emphasis on assistive

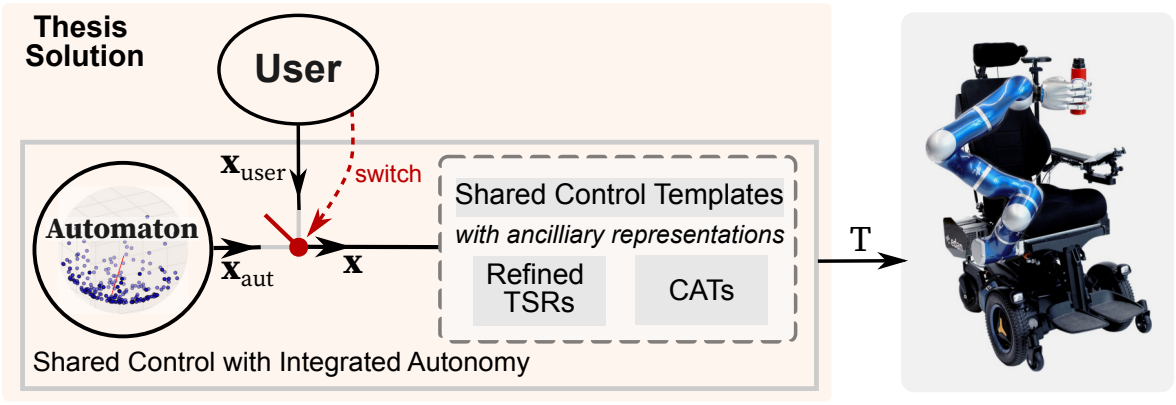



**Figure 7.1:** A simplified diagram of cognitive shared control. A centralized world model, containing geometric and symbolic information of the world state, informs all modules. Based on Shared Control Templates [155], the user's low-dimensional commands from a 3D joystick, $x_{user}$, are processed in a state machine using input mappings and active constraints. Its output is a 6D pose $\mathbf{T}$ that is sent to the robot controller. Four methods are highlighted: (**A**) An Automaton algorithm from Chapter 3, which issues commands $x_{aut}$ on the same space as the human input, and thus can control Shared control Templates. (**B**) A set of planning algorithms with geometric feasibility checks (Chapter 4) and symbolic task planning (Chapter 5). (**C**) Two complementary representations, the refined Task Space Regions (Chapter 4) and the Constraint-Action Templates (Chapter 5), inform the planning algorithms. (**D**) A method for natural language question answering that enables a user to ask questions about the robot's state and beliefs.

robotics. This survey takes the form of a taxonomy, highlighting the various types of support a robot can provide to users while performing everyday tasks. By explicitly understanding the types of shared control, this chapter provides a theoretical framework to ground the methods of the thesis.

The thesis is based on two pre-existing knowledge representation methods, also introduced in **Chapter 2**. First, the ***World State Representation*** (WSR) world model provides the robot with geometric and symbolic information of the state of the world, some of which is obtained with perception algorithms. Its static object database stores object properties and frames of known objects of interest. Second, the ***Shared Control Templates*** (SCT) represent the robot behavior in shared control. They process user commands to achieve a robot pose, utilizing input mappings and active constraints. The SCTs were also complemented with two action representations described in the individual chapters.

The thesis then introduces methods for realizing the four desired features of cognitive shared control. The first one provides **flexibility** and is presented on **Chapter 3**. The SCTs are extended with a method for seamlessly transitioning between shared control and supervised autonomy. That way, users can decide at any point how to execute a given task. They can decide whether to pour a drink themselves or let the robot do it while they lie comfortably in bed. The key technical advancement is the Automaton, an algorithm that produces user commands in the same input space as users. As everything happens inside the SCTs, transitions are seamless.

**Chapter 4** provides a concept for **robustness** using feasibility checks. By leveraging

knowledge of the environment and robot kinematics, the robot can assess the feasibility of a task before it is even started, preventing users from initiating infeasible tasks. Moreover, by computing Refined Task Space Regions (Refined TSRs), the first of the newly-introduced *complementary representations*, the robot provides users with so-called refined constraints that guarantee freedom of movement in a "tested zone" that precludes any kinematic limitations.

The geometric reasoning is extended to the symbolic domain in **Chapter 5**, which presents a task planning algorithm that sequences shared control actions into a goal-directed SCT, providing users with the **goal-guidance** feature. To ease the task planning procedure, the chapter introduces a second complementary representation: the Constraint Action Templates (CATs). CATs leverage declarative knowledge (in the form of PDDL) to enable symbolic planning, as well as geometric knowledge in the form of operations.

Finally, it is essential that users maintain a clear understanding of the robot's state and its internal beliefs. **Explainability** is achieved in **Chapter 6** with RACCOON, a module that enables users to ask questions and general queries to a robot. RACCOON leverages an LLM and so-called "*state summaries*" retrieved from the robot modules to provide explanations grounded on up-to-date robot states, while maintaining the flexibility and open-vocabulary characteristics of foundation models.

While these features are presented in four different papers (and thus presented to the community in isolation from one another), Fig. 7.1 summarizes them within a joint framework. As the SCTs are the main action representation, users still do not require any replanning to switch between representations — symbolic and geometric planning occur only once at the start[1]. After the robot creates a symbolic and a geometric plan, users can seamlessly switch between shared control and supervised autonomy at any time, without needing to create new plans.

## 7.2 Derived works

RACCOON introduced *state summaries*, an interface for LLMs to interact with cognitive robots, achieving explainability. There have been two derived works that extend the state summaries paradigm [2]:

- Besch et. al. [13] introduces a method for explainability and collaborative recovery in the context of EDAN. The framework is explicitly framed for so-called contrastive

[1] As noted in the time benchmarks in Chapters 4 and 5, generating task plans and feasibility checks takes only one or two seconds each.

[2] **Contribution note:** The methods of these two works belong to their first authors, where I provided a supervisory role. The Model Reconciliation method [13] is an extension of Britt Besch's original bachelor's thesis [23]. NealAI [147]'s contribution belongs to Philipp Knestel's ongoing doctoral studies.

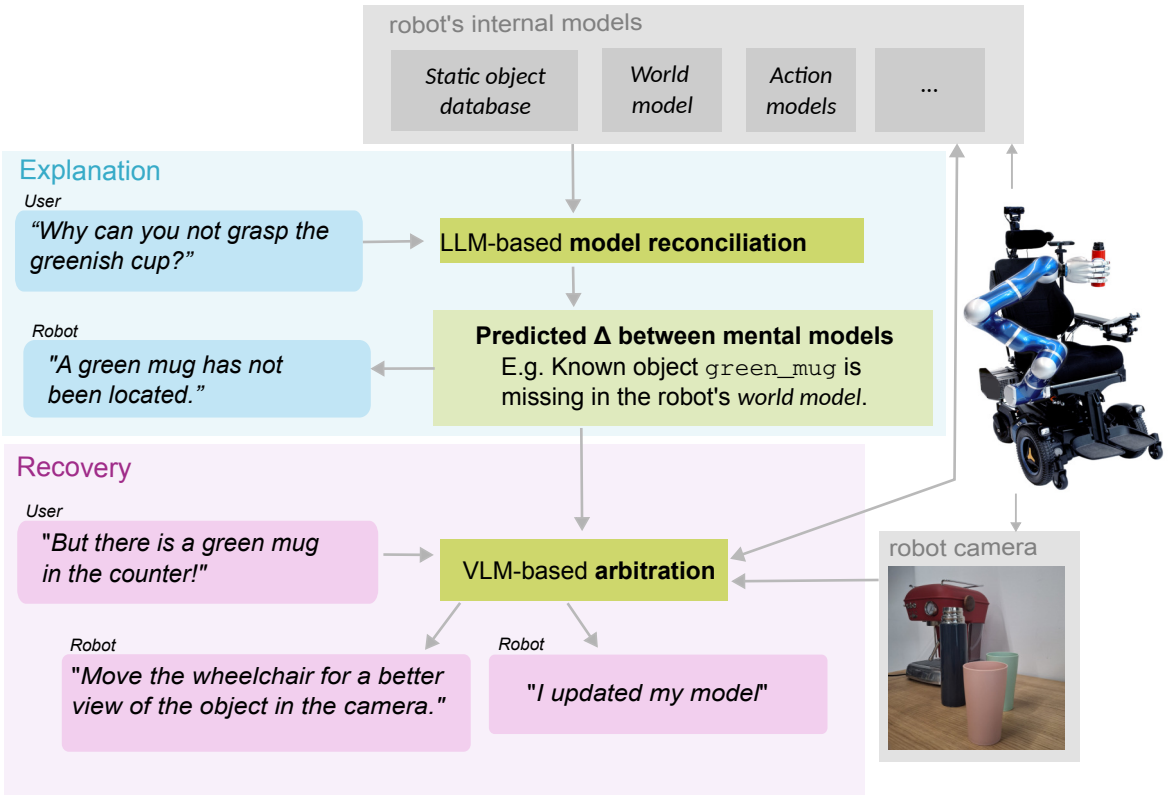



**Figure 7.2:** A framework for model reconciliation through explainability and collaborative recovery, where an LLM uses state summaries from the robot's internal models to provide the difference between human and user mental models. Credits: Taken from [13], under review at the time of submission.

questions of the type "`why cannot you perform action X`", and instead of searching using a vector store as with RACCOON, it formalizes the query as a model reconciliation process. As summarized in Fig. 7.2, the method employs an LLM workflow to iterate over different robot modules (and their corresponding *state summaries*) and predict the difference between the human and robot mental models. It then uses this information to recover from the situation, for instance, by enabling users to update the robot model.

- Knestel et. al. [147] introduces NealAI, a RAG-based chat assistant supporting command of a robotic team from the International Space Station during the Surface Avatar space mission in July 2025, shown in Fig. 7.3. The mission involves an ISS crew member controlling a heterogeneous team of robots that perform ground tasks for environmental investigation and sample return at the German Space Operations Center (GSOC). NealAI utilizes a tool-based approach [174, 152] to retrieve a set of state summaries from the robot team, enabling it to answer open-ended questions from the astronaut. In the example in the figure, the crew member asks for instructions to retrieve a sample container, which retrieves a state summary from Rollin' Justin's task model.




**Figure 7.3:** NealAI deployed for the Surface Avatar mission. The image shows a real interaction with an ISS crew member who was teleoperating Rollin' Justin. NealAI collected a state summary from Rollin' Justin, which contained instructions to pick up a sample container, requiring localization and navigation to the sample container's handover station. It is noted that Surface Avatar features three other robotic assets, the quadrupeds Bert and Spot, and ESA's Interact rover. The robots were performing tasks in a simulated Martian environment containing various props, including a Lander, a solar power unit, a sample return unit, and a large rock with a cave. **Credits:** Figure taken from [147]. © 2025 IEEE.

## 7.3 Discussion

The thesis aims to demonstrate that enabling the robot to acquire and interpret knowledge provides shared control users with four desired features: **flexibility**, **robustness**, **goal-guidance**, and **explainability**. These features were demonstrated experimentally through various activities of daily living using the wheelchair-based mobile manipulator EDAN, as well as the humanoid robot Rollin' Justin.

As the primary focus of this thesis is the methodological advancements required to achieve these features, the impact of the methods has been quantified in several empirical evaluations with the robots. We have measured, for instance, the success rate of seamless transitions between shared control and supervised autonomy (Section 3.3), the accuracy of feasibility check predictions before shared control tasks (Section 4.3), the ability to create task plans with several shared control actions (Section 5.4), and the truthfulness of robot explanations (Section 6.5). A subsequent step is to evaluate the impact of these methods on the user experience, interface design, and overall system usability. Conducting large-scale user studies to determine how users perceive the methods is beyond the scope of this thesis, which focuses on methodological advancements.

As the thesis includes a large technical implementation with heterogeneous components, one key aspect is that there are several transversal contributions from sub-fields of robotics and machine learning, in the context of shared control. These fields are ***knowledge Representation and Reasoning***; ***Teleoperation***; ***Motion Planning***, concerning RRT, Jacobian (for

the feasibility checks), and local optimization-based [3] plans (for the Automaton); ***Task Planning***, for instance with PDDL symbols; **Foundation Models** such as LLMs and VLMs [4]; and **Retrieval-Augmented Generation**.

This broad exploration brought many lessons learned. A particular takeaway from using an optimization-based algorithm for the Automaton is that approaches that simplify (shared) control for human users also enable easier command generation for algorithms. The Automaton is a simple greedy algorithm, and the feasibility of using it in this context is a key feature of the shared control framework.

Another significant insight is that shared control methods can involve the context of the robot as given by the symbolic and geometric information in the world model. As has been shown, this context enables the robot to plan tasks ahead and also prevents users from initiating unfeasible tasks. Questions remain on the user interface, particularly concerning recovery from unfeasible situations. There are two particular avenues to improve these approaches: First, reducing the processing time of feasibility checking algorithms and planning can be achieved by employing new strategies, such as utilizing capability maps [149] to determine kinematic reachability instead of Jacobian-based kinematic methods. And second, creating language-based interfaces based on the reconciliation of mental models, as exemplarized in Besch. et. al. [13]. The methods in that paper can be further extended to handle recovery situations where feasibility checks have failed.

Finally, state summaries enable users to interact with robots, using LLM workflows as intermediaries. The key element is that LLMs and VLMs excel at processing general information, even if it has complex robotic jargon. They also excel at instruction-following, which helps process information in a specific way or for directing the robot to answer with a specific persona (and creating the gimmick of it impersonating the robot). As shown in the two derived works [147, 13], this opened new research directions in explainability, even beyond assistive robotics in a space domain. However, future efforts need to be undertaken concerning reducing prompt engineering efforts, where steps such as automatic prompt optimization [7, 100] and agentic language models [67] can be helpful. Explainability efforts also need to be combined with methods for *commanding* the robot, i.e., providing a goal in natural language instead of in PDDL as in Chapter 5.

## 7.4 Open research directions and current trends

Foundation models drastically improved the generalization capabilities of Artificial Intelligence systems. Although Fig. 6.7 may imply it, our work in explainability (where we benchmark against a “naive VLM”) is not an argument against the use of VLMs in robotics. Instead,

[3] Akin to model-predictive control.

[4] particularly relatively small local models running on a GPU assigned to the robot.

the goal is to combine foundation models with cognitive robot architectures to exploit the best from both worlds. This section provides a small survey of novel research directions and current trends, focused on two axes: representation and acquisition of knowledge ( Section 7.4.1), and novel interaction paradigms for shared control in assistive robotics (Section 7.4.2).

### 7.4.1 Foundation model-powered knowledge acquisition and representation

In the few years since the advent of foundation models, a substantial number of methods have sought to integrate them with cognitive robotic architectures. For instance, VLMs can generate action representations such as task affordances [26] and manipulation constraints based on keypoints [84].

Novel state-of-the-art vision encoders, such as DINO [141, 182] or SIGLIP [206], provide generalist feature priors that can be used to detect objects and retrieve geometric information, informing a world model with geometric information. Some recent 3D foundation models even provide 6D pose estimation of previously-unknown objects [113]. Other methods focus on obtaining symbolic information, such as symbolic predicates that describe the world and the robot's state [45, 98, 83]. Novel, smaller VLMs [132, 200] can further support this task by answering open-ended questions.

### 7.4.2 Advanced interaction paradigms for shared control in assistive robotics

Modern AI methods can be leveraged to enhance interaction paradigms for assistive robots. In one example, a VLM-based autonomous feeding system orchestrates a set of motion primitives (e.g., push food, twirl spaghetti) to feed a user, taking verbal user preferences or instructions (such as *"do not feed me meatballs"*) into account [91] Another example closer to the shared control literature is Autonomous Mode Switching. It consists of using an LLM to propose mode switches automatically, based on task context, and given a language instruction. That way, if the robot is pouring, an LLM can switch the user-controlled DoF automatically from an EE translation to a rotation [189], instead of relying on pre-coded sequences. That way, an LLM helps by automatically deciding which control variables are controlled by users. Further research can focus on increasing the interactivity of such methods.

The future perspective in command blending work is to incorporate more intelligent and task-aware autonomy. To illustrate, Nakamura et al. ([137]) have recently developed a so-called *safety filter* for teleoperation. Although not directly using the formalism in Eq. (2.3), this method allows for the robot to override the teleoperator when it deems an action is unsafe (i.e, causing an undesired event), as estimated via an anomaly detector model based on a forward world model built on DINO [141]. For example, suppose the forward world models predict that moving in a specific direction will cause spillage while manipulating a

container. In that case, it can prevent commands in that direction of motion, thus avoiding an incident. An important aspect to consider is how to make these automatic corrections explainable, as they need to be communicated to users to avoid them feeling out of control.

## 7.5 Closing remarks

The question *"How can the robot provide support to users while empowering them to be in control of the robot?"* is extensively analyzed in the context of shared control. After surveying different perspectives from the literature, the thesis introduces a set of shared control methods that interpret and understand the environment to provide said support, focusing on four pillars: **flexibility** on the amount of control they can have; **robustness** during shared control tasks; **guidance** towards a specific **goal**; and **explainability** of the robotic behavior. These have been implemented on real-world assistive robots. Therefore, the resulting ***cognitive*** shared control paradigm has the potential to improve the quality of life of people with motor impairments. Future efforts will focus on understanding the methods and how users perceive them, with the ultimate goal of deploying them in real households to positively impact the quality of life of target users.

# Bibliography


[1] Eric Jang 1X company Jack Monas. *1x World Model*. September 2024. URL: https://www.1x.tech/discover/1x-world-model.

[2] A. Dragan and S Srinivasa. "A policy-blending formalism for shared control". en. In: *The International Journal of Robotics Research* 32.7 (June 2013), pp. 790–805. ISSN: 0278-3649, 1741-3176. (Visited on 02/17/2020).

[3] D. Aarno, S. Ekvall, and D. Kragic. "Adaptive Virtual Fixtures for Machine-Assisted Teleoperation Tasks". In: *Proceedings of the 2005 IEEE ICRA*. 2005, pp. 1139–1144. URL: 10.1109/ROBOT.2005.1570269.

[4] David A. Abbink, Mark Mulder, and Erwin R. Boer. "Haptic shared control: smoothly shifting control authority?" In: *Cognition, Technology & Work* 14.1 (Nov. 2011), pp. 19–28. URL: 10.1007/s10111-011-0192-5.

[5] Firas Abi-Farraj, Nicolò Pedemonte, and Paolo Robuffo Giordano. "A visual-based shared control architecture for remote telemanipulation". In: *IEEE/RSJ IROS 2016*. 2016, pp. 4266–4273. URL: 10.1109/IROS.2016.7759628.

[6] E. Aertbeliën and J. De Schutter. "Etasl/eTC: A Constraint-Based Task Specification Language and Robot Controller Using Expression Graphs". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1540–1546. DOI: 10.1109/IROS.2014.6942760.

[7] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. *GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning*. 2025. arXiv: 2507.19457 [cs.CL]. URL: https://arxiv.org/abs/2507.19457.

[8] Brian Armour, Elizabeth Courtney-Long, Michael Fox, Heidi Fredine, and Anthony Cahill. "Prevalence and Causes of Paralysis-United States, 2013". In: *American journal of public health* 106 (Aug. 2016). DOI: 10.2105/AJPH.2016.303270.

[9] Reza Azadeh, Muhammad Rana, and Sonia Chernova. “Generalized Cylinders for Learning, Reproduction, Generalization, and Refinement of Robot Skills”. In: July 2017. DOI: 10.15607/RSS.2017.XIII.021.

[10] Sebastian Bader. “Modeling and learning of building blocks for shared control templates”. MA thesis. Technische Universität München, 2021.

[11] Ribin Balachandran, Hrishik Mishra, Matteo Cappelli, Bernhard Weber, Cristian Secchi, Christian Ott, and Alin Albu-Schaeffer. “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. DOI: 10.1109/ICRA40945.2020.9196941.

[12] Georg Bartels, Ingo Kresse, and Michael Beetz. “Constraint-based movement representation grounded in geometric features”. en. In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Atlanta, GA: IEEE, Oct. 2013, pp. 547–554. ISBN: 978-1-4799-2617-6 978-1-4799-2619-0. URL: http://ieeexplore.ieee.org/document/7030027/ (visited on 08/06/2019).

[13] B.Besch, T.Mai, J.Thun, M.Huff, J.Vogel, F.Stulp, and S.Bustamante. *Model Reconciliation through Explainability and Collaborative Recovery in Assistive Robotics*. Under review for the 2026 IEEE International Conference on Robotics and Automation (ICRA), Vienna, Austria. 2025.

[14] Patrick Bechon, C. Lesire, and M. Barbier. “Hybrid planning and distributed iterative repair for multi-robot missions with communication losses”. In: *Autonomous Robots* 44 (2020), pp. 505–531.

[15] Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoğlu, and Georg Bartels. “Know Rob 2.0 — A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 512–519. DOI: 10.1109/ICRA.2018.8460964.

[16] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mosenlechner, Dejan Pangercic, Thomas Ruhr, and Moritz Tenorth. “Robotic roommates making pancakes”. en. In: *2011 11th IEEE-RAS International Conference on Humanoid Robots*. Bled: IEEE, Oct. 2011, pp. 529–536. ISBN: 978-1-61284-866-2 978-1-61284-868-6. URL: http://ieeexplore.ieee.org/document/6100855/ (visited on 11/12/2019).

[17] M. Behery. *A Knowledge-Based Activity Representation for Shared Autonomy Teleoperation of Robotic Arms*. Master’s Thesis. RWTH-Aachen University, 2016.

[18] Khaled Belghith, Benjamin Auder, Froduald Kabanza, Philipe Bellefeuille, and Leo Hartman. “Automatic Animation Generation of a Teleoperated Robot Arm”. In: *ECAI 2008*. IOS Press, 2008, pp. 931–932.

[19] Atharv Belsare, Zohre Karimi, Connor Mattson, and Daniel S. Brown. *Toward Zero-Shot User Intent Recognition in Shared Autonomy*. 2025. arXiv: 2501.08389 [cs.RO]. URL: https://arxiv.org/abs/2501.08389.

[20] "Dmitry Berenson, Siddhartha Srinivasa, Dave Ferguson, and James Kuffner". "Manipulation Planning on Constraint Manifolds". In: (May 2009). URL: 10.1184/R1/6555476.v1.

[21] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. "Task Space Regions: A framework for pose-constrained manipulation planning". In: *The International Journal of Robotics Research* 30.12 (2011), pp. 1435–1460. eprint: https://doi.org/10.1177/0278364910396389. URL: 10.1177/0278364910396389.

[22] Dmitry Berenson, Siddhartha S. Srinivasa, Dave Ferguson, and James J. Kuffner. "Manipulation planning on constraint manifolds". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 625–632. DOI: 10.1109/ROBOT.2009.5152399.

[23] Britt Besch. "A Model Reconciliation Approach for Collaborative Recovery Behavior in Explainable Shared Control". Bachelor's. Ludwig-Maximilians-Universität München, 2024. URL: https://elib.dlr.de/207196/.

[24] Britt Besch, Jeremias Thun, Tai Mai, Markus Huff, Jörn Vogel, and Freek Stulp. "A Model Reconciliation Approach for Collaborative Recovery Behavior in Explainable Shared Control". In: *Under review for the 2026 IEEE International Conference on Robotics and Automation (ICRA)*. 2026.

[25] 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". en. In: *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. Cambridge United Kingdom: ACM, Mar. 2020, pp. 181–190. ISBN: 978-1-4503-6746-2. URL: https://dl.acm.org/URL/10.1145/3319502.3374818 (visited on 09/10/2020).

[26] Timo Birr, Christoph Pohl, Abdelrahman Younes, and Tamim Asfour. *AutoGPT+P: Affordance-based Task Planning with Large Language Models*. 2024. arXiv: 2402.10778 [cs.RO].

[27] Rita Borgo, Michael Cashmore, and Daniele Magazzeni. *Towards Providing Explanations for AI Planner Decisions*. 2018. arXiv: 1810.06338 [cs.AI].

[28] Christoph Borst, Thomas Wimböck, Florian Schmidt, Matthias Fuchs, Bernhard Brunner, Franziska Zacharias, Robuffo-Giordano Paolo, Rainer Konietschke, Wolfgang Sepp, Stefan Fuchs, Christian Rink, Alin Olimpiu Albu-Schäffer, and Gerd Hirzinger. "Rollin'

Justin - Mobile Platform with Variable Base". In: *Best Video*. 2009. URL: https://elib.dlr.de/62526/.

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

[30] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[31] H. Bruyninckx and J. De Schutter. "Specification of force-controlled actions in the "task frame formalism"-a synthesis". In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 581–589. URL: 10.1109/70.508440.

[32] Herman Bruyninckx. *Building Blocks for Complicated and Situational Aware Robotic and Cyber-Physical Systems with Common Sense*. https://robmosys.pages.gitlab.kuleuven.be/. Manuscript version: f600cf - Tuesday 9th January, 2024 08:29. 2024.

[33] Daniel Bullock, Stephen Grossberg, and Frank H. Guenther. "A Self-Organizing Neural Model of Motor Equivalent Reaching and Tool Use by a Multijoint Arm". en. In: *Journal of Cognitive Neuroscience* 5.4 (Oct. 1993), pp. 408–435. ISSN: 0898-929X, 1530-8898. URL: http://www.mitpressjournals.org/URL/10.1162/jocn.1993.5.4.408 (visited on 05/19/2020).

[34] S.R. Buss. *Introduction to Inverse Kinematics with Jacobian Transpose, PseuURLnverse and Damped Least Squares methods*. Tech. rep. University of California, San Diego, 2009.

[35] Samuel Bustamante, Markus Knauer, Jeremias Thun, Stefan Schneyer, Alin Albu-Schäffer, Bernhard Weber, and Freek Stulp. "RACCOON: Grounding Embodied Question-Answering with State Summaries from Existing Robot Modules". In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. 2025, pp. 4322–4329. DOI: 10.1109/ICRA55743.2025.11127843.

[36] 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. URL: 10.1109/LRA.2021.3064449.

[37] 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)*. 2022, pp. 3775–3782. DOI: 10.1109/ICRA46639.2022.9811360.

[38] Samuel Bustamante, Ismael Rodríguez, Gabriel Quere, Peter Lehner, Maged Iskandar, Daniel Leidner, Andreas Dömel, Alin Albu-Schäffer, Jörn Vogel, and Freek Stulp. “Feasibility Checking and Constraint Refinement for Shared Control in Assistive Robotics”. In: *IEEE Robotics and Automation Letters* 9.9 (2024), pp. 8019–8026. DOI: 10.1109/LRA.2024.3430710.

[39] Alexandre Campeau-Lecours and Clément Gosselin. “An anticipative kinematic limitation avoidance algorithm for collaborative robots: Two-dimensional case”. In: *IROS 2016*. 2016, pp. 4232–4237. URL: 10.1109/IROS.2016.7759623.

[40] Shuangyuan Cao and Fang Liu. “A Project-Based Study on User Guidance for Interaction Design”. In: *Cross-Cultural Design. Experience and Product Design Across Cultures. HCII 2021*. Ed. by Pei-Luen Patrick Rau. Vol. 12771. Lecture Notes in Computer Science. Cham: Springer, 2021, pp. 3–13. DOI: 10.1007/978-3-030-77074-7_1. URL: https://doi.org/10.1007/978-3-030-77074-7_1.

[41] Antonio L. Carrillo and Juan A. Falgueras. “Proposal and testing goals-guided interaction for occasional users”. In: *Human-centric Computing and Information Sciences* 10.1 (2020). URL: 10.1186/s13673-020-0209-2.

[42] German Aerospace Center, Gabriel Quere, Maged Iskandar, Simon Spielmann, Miriam Welser, Samuel Bustamante, Sebastian Jung, Felix Schiel, Annette Hagengruber, and Jörn Vogel. *Assistive Robotic Manipulation with Scalable Autonomy*. https://youtu.be/EMs0RL8Ppxw?feature=shared&t=121. 2024.

[43] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. “Plan Explanations as Model Reconciliation: Moving beyond Explanation as Soliloquy”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI’17. Melbourne, Australia: AAAI Press, 2017, 156 to 163. ISBN: 9780999241103.

[44] Jessie Y.C. Chen, Michael Boyce, Julia Wright, and Michael Barnes. *Situation Awareness-Based Agent Transparency*. Tech. rep. ARL-TR-6905. Approved for public release; distribution is unlimited. Aberdeen Proving Ground, MD 21005-5425: Army Research Laboratory, 2014.

[45] Siwei Chen, Anxing Xiao, and David Hsu. *LLM-State: Open World State Representation for Long-horizon Task Planning with Large Language Model*. 2024. arXiv: 2311.17406 [cs.RO]. URL: https://arxiv.org/abs/2311.17406.

[46] Jonathan Claassens and Yiannis Demiris. “Exploiting affordance symmetries for task reproduction planning”. In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. 2012, pp. 653–659. URL: 10.1109/HUMANOIDS.2012.6651589.

[47] Samuel T. Clanton. “Brain-Computer Interface Control of an Anthropomorphic Robotic Arm”. PhD thesis. Pittsburgh, PA 15213: Carnegie Mellon University, 2011. URL: http://www.ri.cmu.edu/publications/techreports/CMU-RI-TR-11-21.pdf.

[48] Samuel T Clanton, Robert G Rasmussen, Zohny Zohny, and Meel Velliste. “Generalized Virtual Fixtures for Shared-Control Grasping in Brain-Machine Interfaces”. In: *Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems* 2013 (2013), pp. 323–328. DOI: 10.1109/IROS.2013.6696371.

[49] Justin L. Collinger, Brian Wodlinger, Jerry E. Downey, Wei Wang, Elizabeth C. Tyler-Kabara, Daniel J. Weber, Andrew J. McMorland, Michele Velliste, Michael L. Boninger, and Arjen B. Schwartz. “High-performance neuroprosthetic control by an individual with tetraplegia”. In: *The Lancet* 381.9866 (2013), pp. 557–564. ISSN: 0140-6736. DOI: 10.1016/S0140-6736(12)61816-9. URL: https://doi.org/10.1016/S0140-6736(12)61816-9.

[50] Coqui. *XTTS-v2: Multilingual Voice Cloning*. 2023. URL: https://huggingface.co/coqui/XTTS-v2.

[51] Chad DeChant, Iretiayo Akinola, and Daniel Bauer. “Learning to Summarize and Answer Questions about a Virtual Robot’s Past Actions”. In: *Autonomous Robots* 47 (2023).

[52] Inderjit S. Dhillon and Suvrit Sra. *Modeling Data using Directional Distributions*. Tech. rep. The University of Texas at Austin, 2003.

[53] DLR-RM. *Mattias and EDAN winning at CYBATHLON Challenges March 2023*. https://www.youtube.com/watch?v=EoER_5vYZsU. 2023.

[54] dscripka. *openWakeWord: An Open-Source Wake-Word Detection Framework*. https://github.com/dscripka/openWakeWord. Version 0.6.0. 2024.

[55] Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. “A tale of two explanations: Enhancing human trust by explaining robot behavior”. In: *Science Robotics* 4.37 (2019), eaay4663. DOI: 10.1126/scirobotics.aay4663. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.aay4663. URL: https://www.science.org/doi/abs/10.1126/scirobotics.aay4663.

[56] Rebecca Eifler, Michael Cashmore, Jörg Hoffmann, Daniele Magazzeni, and Marcel Steinmetz. “A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.06 (2020), pp. 9818–9826. DOI: 10.1609/aaai.v34i06.6534. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6534.

[57] Jianxiang Feng, Maximilian Durner, Zoltan-Csaba Marton, Ferenc Balint-Benczedi, and Rudolph Triebel. *Introspective Robot Perception using Smoothed Predictions from Bayesian Neural Networks*. 2021. arXiv: 2109.12869 [cs.RO]. URL: https://arxiv.org/abs/2109.12869.

[58] Figure. *Figure Status Update - OpenAI Speech-to-Speech Reasoning*. https://www.youtube.com/watch?v=Sq1QZB5baNw. 2024.

[59] E. Fraenkel. *Kent distribution*. 2017. URL: https://github.com/edfraenkel/kent_distribution.

[60] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2312.10997 [cs.CL].

[61] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. “Integrated Task and Motion Planning”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), pp. 265–293. eprint: https://doi.org/10.1146/annurev-control-091420-084139. URL: 10.1146/annurev-control-091420-084139.

[62] Roland Geraerts and Mark H. Overmars. “Creating High-quality Paths for Motion Planning”. In: *The International Journal of Robotics Research* 26.8 (2007), pp. 845–863. URL: 10.1177/0278364907079280.

[63] E. Amir M. Ghalamzan, Firas Abi-Farraj, Paolo Robuffo Giordano, and Rustam Stolkin. “Human-in-the-loop optimisation: Mixed initiative grasping for optimally facilitating post-grasp manipulative actions”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 3386–3393. URL: 10.1109/IROS.2017.8206178.

[64] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. *PDDL, The Planning Version Domain Definition Language. Version 1.2*. Tech. rep. AIPS-98 Planning Competition Committee: Yale Center for Computational Vision and Control, 1998.

[65] James J Gibson. *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin Harcourt (HMH), 1979, p. 127.

[66] Deepak Gopinath, Siddarth Jain, and Brenna D. Argall. “Human-in-the-Loop Optimization of Shared Autonomy in Assistive Robotics”. In: *IEEE Robotics and Automation Letters* 2.1 (Jan. 2017), pp. 247–254. URL: 10.1109/lra.2016.2593928.

[67] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. *Large Language Model based Multi-Agents: A Survey of Progress and Challenges*. 2024. arXiv: 2402.01680 [cs.CL]. URL: https://arxiv.org/abs/2402.01680.

[68] David Ha and Jürgen Schmidhuber. “Recurrent World Models Facilitate Policy Evolution”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

[69] A. Hagengruber, G. Quere, M. Iskandar, S. Bustamante, J. Feng, D. Leidner, A. Albu-Schäffer, F. Stulp, and J. Vogel. “An assistive robot that enables people with amyotrophia to perform sequences of everyday activities”. In: *Accepted to Scientific Reports* (2025).

[70] 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. DOI: 10.1109/EMBC.2018.8512703.

[71] Janik Hager, Ruben Bauer, Marc Toussaint, and Jim Mainprice. “GraspME - Grasp Manifold Estimator”. In: *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. Vancouver, BC, Canada: IEEE Press, 2021, 626–632. URL: https://doi.org/10.1109/RO-MAN50785.2021.9515479.

[72] Katharina Hagmann, Anja Hellings-Kuss, Julian Klodmann, Rebecca Richter, Freek Stulp, and Daniel Leidner. “A Digital Twin for Contextual Assistance in Surgical Robotics Training”. In: *Frontiers in AI and Robotics* (2021).

[73] Katharina Hagmann, Anja Hellings-Kuß, Julian Klodmann, Rebecca Richter, Freek Stulp, and Daniel Leidner. “A Digital Twin Approach for Contextual Assistance for Surgeons During Surgical Robotics Training”. In: *Frontiers in Robotics and AI* 8 (2021). ISSN: 2296-9144. DOI: 10.3389/frobt.2021.735566. URL: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.735566.

[74] Stephen Hart, Paul Dinh, and Kimberly Hambuchen. “The Affordance Template ROS package for robot task programming”. In: *2015 IEEE ICRA*. 2015, pp. 6227–6234. URL: 10.1109/ICRA.2015.7140073.

[75] S. Hayati and S. T. Venkataraman. “Design and implementation of a robot control system with traded and shared control capability”. In: *Proceedings, 1989 ICRA*. 1989, 1310–1315 vol.3.

[76] Bradley Hayes and Julie A. Shah. “Improving Robot Controller Transparency Through Autonomous Policy Explanation”. In: *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’17. Vienna, Austria: Association for Computing Machinery, 2017, 303–312. ISBN: 9781450343367. DOI: 10.1145/2909824.3020233. URL: https://doi.org/10.1145/2909824.3020233.

[77] M. Helmert. “The Fast Downward Planning System”. In: *J. Artif. Intell. Res.* 26 (2006), pp. 191–246.

[78] Laura V. Herlant, Rachel M. Holladay, and Siddhartha S. Srinivasa. “Assistive teleoperation of robot arms via automatic time-optimal mode switching”. en. In: *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Christchurch, New Zealand: IEEE, Mar. 2016, pp. 35–42. ISBN: 978-1-4673-8370-7. URL: http://ieeexplore.ieee.org/document/7451731/ (visited on 05/19/2020).

[79] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J Bohg, T. Asfour, and S. Schaal. “Learning of grasp selection based on shape-templates”. In: *Autonomous Robots* 36.1-2 (Jan. 2014), pp. 51–65. URL: 10.1007/s10514-013-9366-8.

[80] G. Hirzinger. “The space robot technology experiment ROTEX on spacelab-mission D2”. In: *Proc. Satellite Symposium 4: Columbus Eight (COSY-8) Utilisation of Earth Orbiting Laboratories*. LIDO-Berichtsjahr=1992, 1992, pp. 301–309. URL: https://elib.dlr.de/28062/.

[81] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. “OctoMap: an efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous Robots* 34.3 (2013). URL: 10.1007/s10514-012-9321-0.

[82] Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. *Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions*. 2025. arXiv: 2503.23278 [cs.CR]. URL: https://arxiv.org/abs/2503.23278.

[83] Joy Hsu, Jiayuan Mao, Joshua B. Tenenbaum, and Jiajun Wu. *What’s Left? Concept Grounding with Logic-Enhanced Foundation Models*. 2023. arXiv: 2310.16035 [cs.CV]. URL: https://arxiv.org/abs/2310.16035.

[84] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. *ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation*. 2024. arXiv: 2409.01652 [cs.RO]. URL: https://arxiv.org/abs/2409.01652.

[85] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. *ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation*. 2024. arXiv: 2409.01652 [cs.RO]. URL: https://arxiv.org/abs/2409.01652.

[86] I. Rodríguez et. al. “Iteratively Refined Feasibility Checks in Robotic Assembly Sequence Planning”. en. In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 1416–1423. ISSN: 2377-3766, 2377-3774. (Visited on 04/15/2019).

[87] IEEE. *Frequently Asked Questions Regarding IEEE Permissions (accessed 07.03.2025)*. https://www.ieee.org/content/dam/ieee-org/ieee/web/org/pubs/permissions_faq.pdf. 2013.

[88] Maged Iskandar, Gabriel Quere, Annette Hagengruber, Alexander Dietrich, and Jorn Vogel. “Employing Whole-Body Control in Assistive Robotics”. en. In: *2019 IEEE/RSJ IROS*. Macau, China: IEEE, Nov. 2019, pp. 5643–5650. ISBN: 978-1-72814-004-9. URL: https://ieeexplore.ieee.org/document/8967772/ (visited on 02/17/2020).

[89] J. Vogel et. al. “An Ecosystem for Heterogeneous Robotic Assistants in Caregiving: Core Functionalities and Use Cases”. In: *IEEE Robotics Automation Magazine* 28.3 (2021), pp. 12–28.

[90] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S. Srinivasa, and J. Andrew Bagnell. “Shared autonomy via hindsight optimization for teleoperation and teaming”. en. In: *The International Journal of Robotics Research* 37.7 (June 2018), pp. 717–742. ISSN: 0278-3649, 1741-3176. URL: 10.1177/0278364918776060 (visited on 05/19/2020).

[91] Rajat Kumar Jenamani, Priya Sundaresan, Maram Sakr, Tapomayukh Bhattacharjee, and Dorsa Sadigh. *FLAIR: Feeding via Long-horizon AcquIsition of Realistic dishes*. 2024. arXiv: 2407.07561 [cs.RO]. URL: https://arxiv.org/abs/2407.07561.

[92] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. *Mixtral of Experts*. 2024. arXiv: 2401.04088 [cs.LG].

[93] K. Muelling et. al. “Autonomy infused teleoperation with application to brain computer interface controlled manipulation”. en. In: *Autonomous Robots* 41.6 (Aug. 2017), pp. 1401–1422. ISSN: 0929-5593, 1573-7527. (Visited on 01/24/2020).

[94] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. “STOMP: Stochastic trajectory optimization for motion planning”. en. In: *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 4569–4574. ISBN: 978-1-61284-386-5. URL: http://ieeexplore.ieee.org/document/5980280/ (visited on 01/24/2020).

[95] Siddharth Karamcheti, Megha Srivastava, Percy Liang, and Dorsa Sadigh. *LILA: Language-Informed Latent Actions*. 2021. arXiv: 2111.03205 [cs.RO]. URL: https://arxiv.org/abs/2111.03205.

[96] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. "Dense Passage Retrieval for Open-Domain Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550. URL: https://aclanthology.org/2020.emnlp-main.550.

[97] SIDNEY KATZ. "Assessing Self-maintenance: Activities of Daily Living, Mobility, and Instrumental Activities of Daily Living". In: *Journal of the American Geriatrics Society* 31.12 (1983), pp. 721–727. DOI: https://doi.org/10.1111/j.1532-5415.1983.tb03391.x. eprint: https://agsjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1532-5415.1983.tb03391.x. URL: https://agsjournals.onlinelibrary.wiley.com/doi/abs/10.1111/j.1532-5415.1983.tb03391.x.

[98] Kento Kawaharazuka, Yoshiki Obinata, Naoaki Kanazawa, Kei Okada, and Masayuki Inaba. "Robotic Applications of Pre-Trained Vision-Language Models to Various Recognition Behaviors". In: *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. 2023, pp. 1–8. DOI: 10.1109/Humanoids57100.2023.10375211.

[99] Kento Kawaharazuka, Yoshiki Obinata, Naoaki Kanazawa, Kei Okada, and Masayuki Inaba. *Robotic State Recognition with Image-to-Text Retrieval Task of Pre-Trained Vision-Language Model and Black-Box Optimization*. 2024. arXiv: 2410.22707 [cs.RO]. URL: https://arxiv.org/abs/2410.22707.

[100] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. *DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines*. 2023. arXiv: 2310.03714 [cs.CL]. URL: https://arxiv.org/abs/2310.03714.

[101] Tanvi Khera and Valluvan Rangasamy. "Cognition and Pain: A Review". In: *Frontiers in Psychology* 12 (2021). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2021.673962. URL: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2021.673962.

[102] Dae-Jin Kim, Rebekah Hazlett-Knudsen, Heather Culver-Godfrey, Greta Rucks, Tara Cunningham, David Portee, John Bricout, Zhao Wang, and Aman Behal. "How Autonomy Impacts Performance and Satisfaction: Results From a Study With Spinal Cord Injured Subjects Using an Assistive Robot". en. In: *IEEE Transactions on Systems, Man,*

*and Cybernetics - Part A: Systems and Humans* 42.1 (Jan. 2012), pp. 2–14. ISSN: 1083-4427, 1558-2426. URL: 10.1109/TSMCA.2011.2159589 (visited on 02/17/2020).

[103] Matt Klingensmith, Michael McDonald, Radhika Agrawal, Chris Allum, and Rosalind Shinkle. *Robots That Can Chat*. https://bostondynamics.com/blog/robots-that-can-chat/. Boston Dynamics, 2023.

[104] Kirill Kokorin, Syeda R. Zehra, Jing Mu, Peter Yoo, David B. Grayden, and Sam E. John. "Semi-Autonomous Continuous Robotic Arm Control Using an Augmented Reality Brain-Computer Interface". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 32 (2024), pp. 4098–4108. DOI: 10.1109/TNSRE.2024.3500217.

[105] David Kortenkamp, R Peter Bonasso, Dan Ryan, and Debbie Schreckenghost. "Traded control with autonomous robots as mixed initiative interaction". In: *AAAI Symposium on Mixed Initiative Interaction*. 1997, pp. 89–94.

[106] Benjamin Krarup, Michael Cashmore, Daniele Magazzeni, and Tim Miller. "Model-based contrastive explanations for explainable planning". English. In: *ICAPS 2019 Workshop on Explainable AI Planning (XAIP)*. 29th International Conference on Automated Planning and Scheduling - ICAPS 2019, ICAPS 2019 ; Conference date: 11-07-2019 Through 15-07-2019. July 2019.

[107] D. Kuhner, L.D.J. Fiederer, J. Aldinger, F. Burget, M. Völker, R.T. Schirrmeister, C. Do, J. Boedecker, B. Nebel, T. Ball, and W. Burgard. "A service assistant combining autonomous robotics, flexible goal formulation, and deep-learning-based brain–computer interfacing". In: *Robotics and Autonomous Systems* 116 (2019), pp. 98–113. ISSN: 0921-8890. URL: https://doi.org/10.1016/j.robot.2019.02.015.

[108] Benjamin Kuipers, Edward A Feigenbaum, Peter E Hart, and Nils J Nilsson. "Shakey: from conception to history". In: *Ai Magazine* 38.1 (2017), pp. 88–103.

[109] Mohit Kumar. "Scene Graph generation from Camera Perception for Task and Motion Planning". The paper cannot be made available at the moment because patent issues still have to be clarified. MA thesis. Technical University of Munich, 2022. URL: https://elib.dlr.de/190638/.

[110] Steven M. LaValle and James J. Kuffner Jr. "Rapidly-Exploring Random Trees: Progress and Prospects". In: *Algorithmic and Computational Robotics*. 1st. Iowa State University, A James J. Kuffner, Jr., University of Tokyo, Tokyo, Japan. A K Peters/CRC Press, 2001, p. 5. ISBN: 9780429062421.

[111] Nicholas Lawrance, Robert DeBortoli, Dylan Jones, Seth McCammon, Lauren Milliken, Austin Nicolai, Thane Somers, and Geoffrey Hollinger. "Shared autonomy for low-cost underwater vehicles". In: *Journal of Field Robotics* 36.3 (2019), pp. 495–516. DOI: https://doi.org/10.1002/rob.21835. eprint: https://onlinelibrary.wiley.

com/doi/pdf/10.1002/rob.21835. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21835.

[112] Jongseok Lee, Jianxiang Feng, Matthias Humt, Marcus G. Müller, and Rudolph Triebel. *Trust Your Robots! Predictive Uncertainty Estimation of Neural Networks with Sparse Gaussian Processes*. 2021. arXiv: 2109.09690 [cs.RO]. URL: https://arxiv.org/abs/2109.09690.

[113] Taeyeop Lee, Bowen Wen, Minjun Kang, Gyuree Kang, In So Kweon, and Kuk-Jin Yoon. *Any6D: Model-free 6D Pose Estimation of Novel Objects*. 2025. arXiv: 2503.18673 [cs.CV]. URL: https://arxiv.org/abs/2503.18673.

[114] Peter Lehner. PhD thesis.

[115] Daniel Leidner, Christoph Borst, and Gerd Hirzinger. "Things are made for what they are: Solving manipulation tasks by using functional object classes". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 429–435.

[116] Daniel Leidner, Alexander Dietrich, Florian Schmidt, Christoph Borst, and Alin Albu-Schaffer. "Object-centered hybrid reasoning for whole-body mobile manipulation". en. In: *2014 IEEE ICRA*. Hong Kong, China: IEEE, May 2014, pp. 1828–1835. ISBN: 978-1-4799-3685-4. URL: http://ieeexplore.ieee.org/document/6907099/ (visited on 11/08/2019).

[117] Daniel Sebastian Leidner. *Cognitive Reasoning for Compliant Robot Manipulation*. Vol. 127. Springer Tracts in Advanced Robotics. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-04857-0 978-3-030-04858-7. URL: http://link.springer.com/10.1007/978-3-030-04858-7 (visited on 05/19/2020).

[118] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. "Retrieval-augmented generation for knowledge-intensive NLP tasks". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[119] Neal Yi-Sheng Lii, Peter Schmaus, Daniel Leidner, Thomas Krueger, Jessica Grenouilleau, Aaron Pereira, Angelo Giuliano, Adrian Simon Bauer, Anne Köpken, Florian Samuel Lay, Marco Sewtz, Nicolai Bechtel, Samuel Bustamante Gomez, Maximilian Denninger, Werner Friedl, Jörg Butterfass, Edmundo Ferreira, Andrei Gherghescu, Thibaud Chupin, Emiel den Exter, Levin Gerdes, Michael Panzirsch, Harsimran Singh, Ribin Balachandrand, Thomas Hulin, Thomas Gumpert, Annika Schmidt, Daniel Seidel, Milan Hermann, Maximilian Maier, Robert Burger, Florian Schmidt, Bernhard Weber, Ralph Bayer, Benedikt Pleintinger, Roman Holderried, Pedro Henrique Pavel-

ski, Armin Wedler, Stefan von Dombrowski, Hansjörg Maurer, Martin Görner, Tilo Wüsthoff, Serena Bertone, Thomas Müller, Gerd Söllner, Christian Ehrhardt, Lucia Brunetti, Linda Holl, Mairead Bévan, Robert Muehlbauer, Gianfranco Visentin, and Alin Albu-Schäffer. "Introduction to Surface Avatar: the First Heterogeneous Robotic Team to be Commanded with Scalable Autonomy from the ISS". In: *Proceedings of the International Astronautical Congress, IAC*. Vol. IAC-22. International Astronautical Federation, IAF, 2022. URL: https://elib.dlr.de/189618/.

[120] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. *LLaVA-NeXT: Improved reasoning, OCR, and world knowledge*. 2024. URL: https://llava-vl.github.io/blog/2024-01-30-llava-next/.

[121] AI @ Meta Llama Team. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[122] Ana Lopes, Joao Rodrigues, Jorge Perdigao, Gabriel Pires, and Urbano Nunes. "A New Hybrid Motion Planner: Applied in a Brain-Actuated Robotic Wheelchair". In: *IEEE Robotics Automation Magazine* 23.4 (2016), pp. 82–93. URL: 10.1109/MRA.2016.2605403.

[123] 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*. 2021. arXiv: 2107.02907 [cs.RO]. URL: https://arxiv.org/abs/2107.02907.

[124] 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.

[125] Daniel Machlab and Rick Battle. *LLM In-Context Recall is Prompt Dependent*. 2024. arXiv: 2404.08865 [cs.CL].

[126] Veronique Maheu, Julie Frappier, Philippe Archambault, and François Routhier. "Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities". In: *IEEE ... International Conference on Rehabilitation Robotics : [proceedings]* 2011 (June 2011), p. 5975397. DOI: 10.1109/ICORR.2011.5975397.

[127] Juan Martínez-Moritz, Ismael Rodríguez, Korbinian Nottensteiner, Jean-Pascal Lutze, Peter Lehner, and Máximo A. Roa. "Hybrid Planning System for In-Space Robotic Assembly of Telescopes using Segmented Mirror Tiles". In: *2021 IEEE Aerospace Conference (50100)*. 2021, pp. 1–16. URL: 10.1109/AERO50100.2021.9438399.

[128] Dennis J. McFarland. “Brain-computer interfaces for amyotrophic lateral sclerosis”. In: *Muscle & Nerve* 61.6 (2020), pp. 702–707. DOI: https://doi.org/10.1002/mus.26828. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mus.26828. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mus.26828.

[129] 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. DOI: 10.1109/CDC.2016.7799299.

[130] Joseph E. Mercado, Michael A. Rupp, Jessie Y. C. Chen, Michael J. Barnes, Daniel Barber, and Katelyn Procci. “Intelligent Agent Transparency in Human–Agent Teaming for Multi-UxV Management”. In: *Human Factors* 58.3 (2016). PMID: 26867556, pp. 401–415. DOI: 10.1177/0018720815621206. eprint: https://doi.org/10.1177/0018720815621206. URL: https://doi.org/10.1177/0018720815621206.

[131] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0004370218305988.

[132] Moondream. *Moondream 3 Preview: frontier-level reasoning at a blazing speed*. https://moondream.ai/blog/moondream-3-preview. Accessed: 2025-10-09. Sept. 2025.

[133] Maximilian Sebastian Mühlbauer, Freek Stulp, Alin Olimpiu Albu-Schäffer, and João Silvério. “Mixture of experts on Riemannian manifolds for visual-servoing fixtures”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, Workshop on Probabilistic Robotics in the Age of Deep Learning*. 2022. URL: https://elib.dlr.de/189970/.

[134] Technical University of Munich. *TUM KI Strategie (in German) (accessed 07.03.2025)*. https://mediatum.ub.tum.de/doc/1766632/1766632.pdf. 2024.

[135] Maximilian Mühlbauer, Thomas Hulin, Bernhard Weber, Sylvain Calinon, Freek Stulp, Alin Albu-Schäffer, and João Silvério. “A Probabilistic Approach to Multi-Modal Adaptive Virtual Fixtures”. In: *IEEE Robotics and Automation Letters* 9.6 (2024), pp. 5298–5305. DOI: 10.1109/LRA.2024.3384759.

[136] Meng Na, Bin Yang, and Peifa Jia. “Improved Damped Least Squares Solution with Joint Limits, Joint Weights and Comfortable Criteria for Controlling Human-like Figures”. In: *2008 IEEE Conference on Robotics, Automation and Mechatronics*. 2008, pp. 1090–1095. URL: 10.1109/RAMECH.2008.4681441.

[137] Kensuke Nakamura, Lasse Peters, and Andrea Bajcsy. *Generalizing Safety Beyond Collision-Avoidance via Latent-Space Reachability Analysis*. 2025. arXiv: 2502.00935 [cs.RO]. URL: https://arxiv.org/abs/2502.00935.

[138] Günter Niemeyer, Carsten Preusche, and Gerd Hirzinger. “Telerobotics”. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 741–757. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_32. URL: https://doi.org/10.1007/978-3-540-30301-5_32.

[139] NVIDIA, : Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. *Cosmos World Foundation Model Platform for Physical AI*. 2025. arXiv: 2501.03575 [cs.CV]. URL: https://arxiv.org/abs/2501.03575.

[140] Openai. *openai/whisper-large-v2*. Accessed: 2025-09-16. URL: https://huggingface.co/openai/whisper-large-v2.

[141] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. *DINOv2: Learning Robust Visual Features without Supervision*. 2024. arXiv: 2304.07193 [cs.CV]. URL: https://arxiv.org/abs/2304.07193.

[142] World Health Organization. *Spinal cord injury fact sheet*. Tech. rep. World Health Organization, 2013. URL: https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury.

[143] Jia Pan, Sachin Chitta, and Dinesh Manocha. “FCL: A general purpose library for collision and proximity queries”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3859–3866. URL: 10.1109/ICRA.2012.6225337.

[144] Michael Panzirsch, Marek Sierotowicz, Revanth Prakash, Harsimran Singh, and Christian Ott. “Deflection-Domain Passivity Control of Variable Stiffnesses Based on Potential Energy Reference”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4440–4447. DOI: 10.1109/LRA.2022.3147566.

[145] Claudia Pérez-D’Arpino and Julie A 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.

[146] Betsy Phillips and Hongxin Zhao. “Predictors of Assistive Technology Abandonment”. In: *Assistive technology : the official journal of RESNA* 5 (Feb. 1993), pp. 36–45. DOI: 10.1080/10400435.1993.10132205.

[147] P.Knestel, L.Mayershofer, A.Köpken, N.Batti, F.Lay, T.Bachmann, S.Brinkman, J.Butterfass, E.DenExter, W.Friedl, T.Gumpert, X.Luo, A.Manaparampil, T.Mai, A.Raffin, F.Schmidt, A.Schmidt, D.Seidel, L.Schurmann, N.Wenzinger, B.Weber, A.Albu-Schäffer, A.Bauer, D.Leidner, R.Luz, P.Schmaus, F.Stulp, T.Krüger, S.Bustamante, and N.Lii. “NealAI: a RAG-based chat assistant supporting command of a robotic team from the ISS”. In: *Proceedings of the 2025 International Conference on Space Robotics (iSpaRo)*. Accepted for presentation. N.Lii and S.Bustamante contributed equally in a supervisory role. 2025.

[148] David Porfirio, Mark Roberts, and Laura M. Hiatt. “Goal-Oriented End-User Programming of Robots”. In: *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’24. ACM, Mar. 2024, 582–591. DOI: 10.1145/3610977.3634974. URL: http://dx.doi.org/10.1145/3610977.3634974.

[149] Oliver Porges, Theodoros Stouraitis, Christoph Borst, and Maximo A Roa. “Reachability and capability analysis for manipulation tasks”. In: *ROBOT2013: First Iberian Robotics Conference: Advances in Robotics, Vol. 2*. Springer. 2014, pp. 703–718.

[150] Vitalii Pruks and Jee-Hwan Ryu. “Method for generating real-time interactive virtual fixture for shared teleoperation in unknown environments”. In: *The International Journal of Robotics Research* 41.9-10 (2022), pp. 925–951. DOI: 10.1177/02783649221102980. eprint: https://doi.org/10.1177/02783649221102980. URL: https://doi.org/10.1177/02783649221102980.

[151] Ásgerður Arna Pálsdóttir, Rasmus Leck Kæseler, Stefan Hein Bengtson, Thomas B. Moeslund, and Lotte N. S. Andreasen Struijk. “Remote Tongue-Based Control of a Wheelchair Mounted Assistive Robotic Manipulator and the Effect of Adaptive Semi-Automation”. In: *IEEE Transactions on Biomedical Engineering* 72.1 (2025), pp. 14–23. DOI: 10.1109/TBME.2024.3435837.

[152] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. *ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs*. 2023. arXiv: 2307.16789 [cs.AI]. URL: https://arxiv.org/abs/2307.16789.

[153] Gabriel Quere. "Learning and designing shared control skills from demonstrations for assistive robots". PhD thesis. Institut Polytechnique de Paris, 2025. URL: https://elib.dlr.de/213091/.

[154] 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. URL: 10.1109/IROS51168.2021.9636047.

[155] Gabriel Quere, Annette Hagengruber, Maged Iskandar, Samuel Bustamante, Daniel Leidner, Freek Stulp, and Joern Vogel. "Shared Control Templates for Assistive Robotics". en. In: *2020 IEEE ICRA*. Paris, France, 2020, p. 7.

[156] Gabriel Quere, Maged Iskandar, Simon Spielmann, Miriam Welser, Samuel Bustamante, Sebastian Jung, Felix Schiel, Annette Hagengruber, and Jörn Vogel. *Assistive Robotic Manipulation with Scalable Autonomy*. YouTube video, DLR RM channel, Creative Commons Attribution license. German Aerospace Center (DLR). Sept. 2024. URL: https://www.youtube.com/watch?v=EMs0RL8Ppxw.

[157] Gabriel Quere, Freek Stulp, David Filliat, and Joao Silverio. "A probabilistic approach for learning and adapting shared control skills with the human in the loop". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024. URL: https://example.com/path/to/hal.science.

[158] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. *Robust Speech Recognition via Large-Scale Weak Supervision*. 2022. arXiv: 2212.04356 [eess.AS]. URL: https://arxiv.org/abs/2212.04356.

[159] 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* 42.5 (June 2018). Received 31 December 2016; Accepted 17 October 2017; Published online 23 November 2017, pp. 1037–1051. DOI: 10.1007/s10514-017-9680-7.

[160] Allen Z. Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh. *Explore until Confident: Efficient Exploration for Embodied Question Answering*. 2024. arXiv: 2403.15941 [cs.RO].

[161] Erik Rietveld and Julian Kiverstein. “A Rich Landscape of Affordances”. In: *Ecological Psychology* 26.4 (2014), pp. 325–352. DOI: 10.1080/10407413.2014.958035. URL: https://doi.org/10.1080/10407413.2014.958035.

[162] DLR Institute of Robotics and Mechatronics. *ROTEX (1988–1993)*. Opened in Ocotber 2024. URL: https://www.dlr.de/en/rm/research/robotic-systems/hands/rotex-1988-1993.

[163] Kinova robotics. *Kinova eating device*. Opened in Ocotber 2024. URL: https://assistive.kinovarobotics.com/product/obi-eating-device.

[164] Kinova robotics. *Kinova robotic arm*. Opened in Ocotber 2024. URL: https://assistive.kinovarobotics.com/product/jaco-robotic-arm.

[165] Ismael Rodríguez, Adrian S Bauer, Korbinian Nottensteiner, Daniel Leidner, Gerhard Grunwald, and Máximo A Roa. “Autonomous robot planning system for in-space assembly of reconfigurable structures”. In: *2021 IEEE Aerospace Conference (50100)*. IEEE. 2021, pp. 1–17.

[166] G.R.B.E. Romer, H.J.A. Stuyt, and A. Peters. “Cost-savings and economic benefits due to the assistive robotic manipulator (ARM)”. In: *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.* 2005, pp. 201–204. DOI: 10.1109/ICORR.2005.1501084.

[167] L. B. Rosenberg. *The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments*. Tech. rep. AL/CF-TR-1994-0089. USAF Armstrong Lab., TX, USA, 1992.

[168] L. B. Rosenberg. “Virtual fixtures: Perceptual tools for telerobotic manipulation”. In: *Proc. IEEE Virt. Real. Annual Int. Symp.* Sept. 1993, pp. 76–82.

[169] S. Iregui et. al. “Reconfigurable Constraint-Based Reactive Framework for Assistive Robotics With Adaptable Levels of Autonomy”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7397–7405.

[170] Fatai Sado, Chu Kiong Loo, Wei Shiung Liew, Matthias Kerzel, and Stefan Wermter. *Explainable Goal-Driven Agents and Robots – A Comprehensive Review*. 2022. arXiv: 2004.09705 [cs.RO].

[171] Ryo Sakagami, Andreas Dömel, Peter Lehner, Sebastian Riedel, Sebastian G. Brunner, Alin Albu-Schäffer, and Freek Stulp. “A Tree-Based World Model for Reducing System Complexity in Autonomous Mobile Manipulation”. In: *IEEE Robotics and Automation Letters* 9.7 (2024), pp. 6680–6687. DOI: 10.1109/LRA.2024.3410156.

[172] Ryo Sakagami, Florian S. Lay, Andreas Dömel, Martin J. Schuster, Alin Albu-Schäffer, and Freek Stulp. “Robotic World Models—Conceptualization, Review, and Engineering Best Practices”. In: *Frontiers in Robotics and AI* 10 (2023). DOI: 10.3389/frobt.2023.1253049.

[173] Aroua Hedhili Sbaï, Wided Lejouad Chaari, and Khaled Ghédira. “Intra-agent Explanation Using Temporal and Extended Causal Maps”. In: *Procedia Computer Science* 22 (2013). 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013, pp. 241–249. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.09.100. URL: https://www.sciencedirect.com/science/article/pii/S1877050913008934.

[174] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. *Toolformer: Language Models Can Teach Themselves to Use Tools*. 2023. arXiv: 2302.04761 [cs.CL]. URL: https://arxiv.org/abs/2302.04761.

[175] Lalithkumar Seenivasan, Mobarakol Islam, and Gokul Kannan. “SurgicalGPT: End-to-End Language-Vision GPT for Visual Question Answering in Surgery”. In: *edical Image Computing and Computer Assisted Intervention – MICCAI 2023, Lecture Notes in Computer Science, vol 14228. Springer, Cham*. 2023.

[176] M. Selvaggio, P. Robuffo Giordano, F. Ficuciello, and B. Siciliano. “Passive Task-Prioritized Shared-Control Teleoperation with Haptic Guidance”. In: *IEEE ICRA 2019*. 2019, pp. 430–436. URL: 10.1109/ICRA.2019.8794197.

[177] Mario Selvaggio, Firas Abi-Farraj, Claudio Pacchierotti, Paolo Robuffo Giordano, and Bruno Siciliano. “Haptic-Based Shared-Control Methods for a Dual-Arm System”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4249–4256. URL: 10.1109/LRA.2018.2864353.

[178] Mario Selvaggio, Fei Chen, Boyang Gao, Gennaro Notomista, Francesco Trapani, and Darwin Caldwell. “Vision based virtual fixture generation for teleoperated robotic manipulation”. In: *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*. 2016, pp. 190–195. DOI: 10.1109/ICARM.2016.7606917.

[179] 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. DOI: 10.1109/LRA.2018.2849876.

[180] Mario Selvaggio, Gennaro Notomista, Fei Chen, Boyang Gao, Francesco Trapani, and Darwin Caldwell. “Enhancing bilateral teleoperation using camera-based online virtual fixtures generation”. In: *IEEE/RSJ IROS 2016*. 2016, pp. 1483–1488. URL: 10.1109/IROS.2016.7759241.

[181] Shingo Shimoda, Lorenzo Jamone, Dimitri Ognibene, Takayuki Nagai, Alessandra Sciutti, Alvaro Costa-Garcia, Yohei Oseki, and Tadahiro Taniguchi. “What is the role of the next generation of cognitive robotics?” In: *Advanced Robotics* 36.1-2 (2022),

pp. 3–16. DOI: 10.1080/01691864.2021.2011780. URL: https://doi.org/10.1080/01691864.2021.2011780.

[182] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. *DINOv3*. 2025. arXiv: 2508.10104 [cs.CV]. URL: https://arxiv.org/abs/2508.10104.

[183] Sarath Sreedharan, Siddharth Srivastava, David Smith, and Subbarao Kambhampati. "Why Can't You Do That HAL? Explaining Unsolvability of Planning Tasks". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 1422–1430. DOI: 10.24963/ijcai.2019/197. URL: https://doi.org/10.24963/ijcai.2019/197.

[184] Mohan Sridharan and Ben Meadows. "Towards a Theory of Explanations for Human–Robot Collaboration". In: *KI - Künstliche Intelligenz* 33 (Sept. 2019), pp. 1–12. DOI: 10.1007/s13218-019-00616-y.

[185] John Stein. "Cerebellar forward models to control movement". In: *The Journal of physiology* 587.2 (2009), p. 299. DOI: 10.1113/jphysiol.2008.167627.

[186] Freek Stulp, Adrian Simon Bauer, Samuel Bustamante Gomez, Florian Samuel Lay, Peter Schmaus, and Daniel Leidner. "Explainability and Knowledge Representation in Robotics: The Green Button Challenge". In: *Explainable Logic-Based Knowledge Representation (XLoKR 2020) at the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*. 2020, pp. 123–134.

[187] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. "Implicit 3D Orientation Learning for 6D Object Detection from RGB Images". en. In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Vol. 11210. Cham: Springer International Publishing, 2018, pp. 712–729. ISBN: 978-3-030-01230-4 978-3-030-01231-1. URL: http://link.springer.com/10.1007/978-3-030-01231-1_43 (visited on 05/16/2019).

[188] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, and Rudolph Triebel. "Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection". In: *International Journal of Computer Vision* (2020). URL: https://elib.dlr.de/135549/.

[189] Yiran Tao, Jehan Yang, Dan Ding, and Zackory Erickson. *LAMS: LLM-Driven Automatic Mode Switching for Assistive Teleoperation*. 2025. arXiv: 2501.08558 [cs.RO]. URL: https://arxiv.org/abs/2501.08558.

[190] Thorsten-Voice. "*Thorsten-Voice*" *TTS Model (Neutral) - Tacotron2 DDC*. URL: https://huggingface.co/Thorsten-Voice/Tacotron2-DDC.

[191] Jeremias Thun. "Storing and Predicting User-Specific Action Chains in Assistive Robots". MA thesis. University of Bremen, 2025.

[192] Nikolaus Vahrenkamp, Dmitry Berenson, Tamim Asfour, James Kuffner, and Rüdiger Dillmann. "Humanoid motion planning for dual-arm manipulation and re-grasping tasks". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 2464–2470. URL: 10.1109/IROS.2009.5354625.

[193] Cristian Vergara, Santiago Iregui, Joris De Schutter, and Erwin Aertbeliën. "Generating Reactive Approach Motions Towards Allowable Manifolds using Generalized Trajectories from Demonstrations". In: *IEEE/RSJ IROS 2020*. 2020, pp. 9697–9704. URL: 10.1109/IROS45743.2020.9340808.

[194] Jorn 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)*. Honolulu, HI: IEEE, July 2018, pp. 2136–2141. ISBN: 978-1-5386-3646-6. URL: https://ieeexplore.ieee.org/document/8512689/ (visited on 02/17/2020).

[195] Jörn Vogel, Justin Bayer, and Patrick van der Smagt. "Continuous robot control using surface electromyography of atrophic muscles". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 845–850. URL: 10.1109/IROS.2013.6696449.

[196] Jörn Vogel, Annette Hagengruber, Maged Iskandar, Gabriel Quere, Ulrike Leipscher, Samuel Bustamante, Alexander Dietrich, Hannes Hoeppner, Daniel Leidner, and Alin Albu-Schäffer. "EDAN - an EMG-Controlled Daily Assistant to Help People with Physical Disabilities". en. In: *2020 IEEE/RSJ IROS*. 2020.

[197] Sebastian Wallkötter, Silvia Tulli, Ginevra Castellano, Ana Paiva, and Mohamed Chetouani. "Explainable Embodied Agents Through Social Cues: A Review". In: *J. Hum.-Robot Interact.* 10.3 (2021). DOI: 10.1145/3457188. URL: https://doi.org/10.1145/3457188.

[198] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL].

[199] Roman Weitschat, Alexander Dietrich, and Jörn Vogel. "Online motion generation for mirroring human arm motion". In: *2016 IEEE ICRA*. 2016, pp. 4245–4250. URL: 10.1109/ICRA.2016.7487620.

[200] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. *Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks*. 2023. arXiv: 2311.06242 [cs.CV]. URL: https://arxiv.org/abs/2311.06242.

[201] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. *C-Pack: Packaged Resources To Advance General Chinese Embedding*. 2023. arXiv: 2309.07597 [cs.CL].

[202] Yang Xu, Heng Zhang, Linfeng Cao, Xiaokang Shu, and Dingguo Zhang. "A Shared Control Strategy for Reach and Grasp of Multiple Objects Using Robot Vision and Noninvasive Brain–Computer Interface". In: *IEEE Transactions on Automation Science and Engineering* 19.1 (2022), pp. 360–372. DOI: 10.1109/TASE.2020.3034826.

[203] Takuma Yoneda, Luzhe Sun, Ge Yang, Bradly Stadie, and Matthew Walter. *To the Noise and Back: Diffusion for Shared Autonomy*. 2023. arXiv: 2302.12244 [cs.RO]. URL: https://arxiv.org/abs/2302.12244.

[204] J-Anne Yow, Neha Priyadarshini Garg, and Wei Tech Ang. "Shared Autonomy of a Robotic Manipulator for Grasping Under Human Intent Uncertainty Using POMDPs". In: *IEEE Transactions on Robotics* 40 (2024), pp. 332–350. DOI: 10.1109/TRO.2023.3334631.

[205] Wentao Yuan, Chris Paxton, Karthik Desingh, and Dieter Fox. *SORNet: Spatial Object-Centric Representations for Sequential Manipulation*. 2022. arXiv: 2109.03891 [cs.RO]. URL: https://arxiv.org/abs/2109.03891.

[206] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. *Sigmoid Loss for Language Image Pre-Training*. 2023. arXiv: 2303.15343 [cs.CV]. URL: https://arxiv.org/abs/2303.15343.

[207] Wenchang Zhang, Fuchun Sun, Chunfang Liu, Weihua Su, Chuanqi Tan, and Shaobo Liu. "A hybrid EEG-based BCI for robot grasp controlling". In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2017, pp. 3278–3283. URL: 10.1109/SMC.2017.8123134.

[208] Youcai Zhang, Xinyu Huang, Jinyu Ma, Zhaoyang Li, Zhaochuan Luo, Yanchun Xie, Yuzhuo Qin, Tong Luo, Yaqian Li, Shilong Liu, Yandong Guo, and Lei Zhang. *Recognize Anything: A Strong Image Tagging Model*. 2023. arXiv: 2306.03514 [cs.CV].

[209] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. *DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning*. 2024. arXiv: 2411.04983 [cs.RO]. URL: https://arxiv.org/abs/2411.04983.

[210] Mark Zolotas and Yiannis Demiris. "Towards Explainable Shared Control using Augmented Reality". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 3020–3026. DOI: 10.1109/IROS40897.2019.8968117.

[211] Matt Zucker, Nathan Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M. Dellin, J. Andrew Bagnell, and Siddhartha S. Srinivasa. “CHOMP: Covariant Hamiltonian optimization for motion planning”. en. In: *The International Journal of Robotics Research* 32.9-10 (Aug. 2013), pp. 1164–1193. ISSN: 0278-3649, 1741-3176. URL: http://journals.sagepub.com/URL/10.1177/0278364913488805 (visited on 01/24/2020).

[212] Adrian Łańcucki. *FastPitch: Parallel Text-to-speech with Pitch Prediction*. 2021. arXiv: 2006.06873 [eess.AS]. URL: https://arxiv.org/abs/2006.06873.