UNIVERSITÄT
BAYREUTH

Fluency in Dynamic Human-Robot Teaming with Intention Prediction

Von der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

von

Nico Höllerich
aus Münchberg

1. Gutachter: Prof. Dr. Dominik Henrich
2. Gutachter: Prof. Dr.-Ing. Bernd Kuhlenkötter
3. Gutachter: Prof. Dr.-Ing. Ulrike Thomas

Tag der Einreichung: 12. Februar 2025
Tag des Kolloquiums: 30. April 2025

Abstract

The market share of collaborative robots in the industry continues to grow steadily. However, there is still a need to improve human-robot collaboration further to support the ongoing industrial transformation, enabling robots to take on more tasks and function as true teammates. This requires advancements in both perception and decision-making capabilities. This thesis contributes to two key aspects of achieving seamless human-robot collaboration: perceiving task progress and recognising human intentions, all within a framework of flexible and fluent cooperation.

The starting point involves two empirical studies on human-human teaming to identify communication mechanisms and intention prediction capabilities for assembly tasks. The findings have resulted in the development of a research demonstrator for a cooperative assembly station, with a particular emphasis on the flexible allocation of task steps and the integration of intention prediction.

The prototype hardware includes a robot arm, an assembly station, and a depth camera. Core innovations are the task-state tracking algorithm and the intention prediction component. The task-state tracking algorithm is based on object detection and occlusion data from the camera. It models task execution in terms of the markings of a Petri net. The provided Petri net encodes all possible ways to execute the task in a space-efficient manner. Comprehensive evaluations demonstrate the algorithm's robustness and efficiency in non-deterministic task executions.

The sequence of task steps executed by the human serves as input to the intention prediction module, which predicts the next steps using a neural network with a custom-designed feature space. This feature space encodes spatial information to enable efficient, real-time training for each user interacting with the robot. A comprehensive evaluation compares the accuracy of the module with traditional action prediction approaches using data from user interactions with the system.

Finally, this thesis presents a comprehensive study on the system. Based on intention prediction, different robot behaviours are implemented and evaluated by a user study in terms of fluency and productivity questionnaires. Post-hoc analysis provides insights into the interrelationships and effects of robot behaviour on these measures.

In summary, this thesis contributes technical foundations, empirical evaluations, and motivates further investigations into fluent and flexible human-robot teaming with intention prediction.

Zusammenfassung

Der Marktanteil der installierten kollaborativen Roboter in der Industrie nimmt stetig zu. Dennoch muss die Zusammenarbeit zwischen Mensch und Roboter im Zuge des industriellen Wandels weiter verbessert werden, damit Roboter mehr Aufgaben übernehmen und zu echten Teamkollegen werden können. Beides erfordert eine weitere Verbesserung der Wahrnehmung und Entscheidungsfindung. In der vorliegenden Arbeit werden die Grundpfeiler für eine fließende Mensch-Roboter-Kooperation untersucht. Die angestrebte Kooperation zeichnet sich durch ihre Flexibilität und ihr fließendes Zusammenspiel aus.

Den Ausgangspunkt der Untersuchung bilden zwei empirische Studien zur Mensch-Mensch-Zusammenarbeit, um Kommunikationsmechanismen und Fähigkeiten zur Intentionserkennung bei Montageaufgaben zu identifizieren. Die Resultate der Untersuchung dienten als Grundlage für die Entwicklung eines Forschungsdemonstrators für eine kooperative Montagestation, wobei der Schwerpunkt auf der flexiblen Zuweisung von Aufgabenschritten und der Integration der Intentionserkennung liegt.

Die Hardware des Prototyps umfasst einen Roboterarm, eine Montagestation und eine Tiefenkamera. Zu den wichtigsten Neuerungen zählen der Algorithmus zur Verfolgung des Aufgabenzustands sowie die Komponente zur Intentionserkennung. Der Algorithmus zur Verfolgung des Aufgabenzustands basiert auf der Objekterkennung und den Verdeckungsdaten der Kamera. Er modelliert die Aufgabenausführung in Form von Markierungen in einem Petri-Netz. Das bereitgestellte Petri-Netz kodiert alle möglichen Wege zur Ausführung der Aufgabe in einer platzsparenden Weise. Die Ergebnisse umfassender Evaluierungen belegen die Robustheit und Effizienz des Algorithmus bei der Ausführung nicht-deterministischer Aufgaben.

Die vom Menschen ausgeführte Sequenz von Aufgabenschritten sind Eingabe für die Intentionserkennung. Das Modul prognostiziert die nächsten Schritte des Menschen unter Verwendung eines neuronalen Netzes mit einem manuell erstellten Merkmalsraums, der räumliche Informationen kodiert. Das Training des neuronalen Netzes geschieht während der Ausführung und passgenau für jeden Nutzer. Eine umfassende Evaluierung vergleicht die Genauigkeit dieses Moduls mit klassischen Ansätzen zur Handlungsvorhersage unter Verwendung von Daten aus Benutzerinteraktionen mit dem System.

Schließlich wird in der vorliegenden Arbeit eine umfassende Studie über das Systems vorgestellt. Auf der Grundlage der Absichtsvorhersage werden verschiedene Verhaltensweisen des Roboters implementiert und durch eine Nutzerstudie hinsichtlich fließender

Zusammenarbeit und Produktivität bewertet. Eine Post-hoc-Analyse gibt Aufschluss über die Zusammenhänge und Auswirkungen des Roboterverhaltens auf diese Messgrößen. Zusammenfassend liefert die vorliegende Arbeit technische Grundlagen, empirische Auswertungen und motiviert zu weiteren Untersuchungen zur fließenden und flexiblen Mensch-Roboter Zusammenarbeit mit Intentionsvorhersage.

Contents

CHAPTER 1

Introduction

The manufacturing industry in many Western countries is undergoing a significant change. A shortage of shop-floor workers and the desire for productivity gains drive this change. Yet full automation is often infeasible due to the limited capabilities of the robot. When combining human and robot as a team, each can contribute their unique strength, namely flexibility, dexterity, and handling complex components for the human as well as precision and endurance for the robot. When properly designed, such a team can increase workers' satisfaction and engagement. A robot, however, needs certain capabilities to perceive, act and adapt to the partner in such a team. Against this background, this thesis explores fluent human-robot teaming with intention prediction. Section 1.1 motivates the demand for introducing teams of humans and robots, as well as the associated benefits and challenges. Section 1.2 illustrates the context in which this thesis considers teams of humans and robots. The contextualisation introduces terms from the domain of industrial assembly and human-robot collaboration. Computer vision is also included as an enabling factor for the robot to understand its environment.

Section 1.3 derives the research questions from the core capabilities a robot must have to participate in human-robot teaming.

1.1. Benefits of Human-Robot Teaming in Industrial Manufacturing

The current manufacturing industry is undergoing significant transformations due to demographic changes and advancements. As part of this transformation, companies digitally monitor and increasingly automate their production processes. Those transformations are often subsumed under the umbrella term of Industry 4.0 [1]. *Industry 4.0* was coined in expectation of a fourth industrial revolution following the steam engine, electrification, and digitalisation. At its core are smart factories that integrate techniques for digital monitoring, data mining, and automation [2]. The desired effects are increased productivity, improved quality, and lower production costs. Products should be highly customizable but producible with the same efficiency as in classical mass production. Automation technology and robotics are important drivers for Industry 4.0 [3, 4]. As such, the installations of industrial robots have tripled over the past decade [5].

In contrast to robots, human shop-floor workers haven't been as readily available in recent years. One reason for this is the demographic change in many Western European countries [6]. Furthermore, fixed working shifts, monotony, and sometimes exhaustive working conditions limit the attractiveness of the job. Companies can utilise robots to overcome the lack of workforce, but pose new challenges on their own, such as functional or social challenges. In terms of social challenges, workers' concerns about job loss, about safety, and about the complexity of interacting with robots can impede their implementation [7]. In terms of functional challenges, robots can achieve high precision and throughput, but do not yet reach human competence levels when it comes to perception capabilities and adapting to product variants. To put it more precisely, they are still limited in terms of flexibility, dexterity, and handling complex components [8]. Small and medium-sized enterprises with small lot sizes or high product variance additionally impose the problem that processes can only be standardised to a limited extent [9]. Costs to adapt robot software to fully automate production are then too high, or an adaptation is infeasible with the current state of technology [10]. Thus, their production chains still partly require manual work.

Therefore, a *teaming approach* where human and robot work together is essential for leveraging the strengths of both partners. Those teaming approaches require decision-

Figure 1.1.: Human and robot work hand in hand to accomplish the packing task.

makers to consider aspects of legislation, safety, productivity, and physical and cognitive ergonomics. Recent prototypes of humanoid robots (e.g. [11, 12, 13]) show the future potential of how robots can reliably and accurately perceive and manipulate objects. This lowers the functional barriers of introducing human-robot teams. To form human-robot teams, the robot must not only have the functional capabilities of doing its part of the task but also the skills required to act in a team, as elaborated in Section 1.3. The goal of human-robot teaming is to utilise the human's dexterity and agency in problem detection. Working in a team can moreover increase workers' satisfaction and engagement (Figure 1.1). For instance, job rotation is a common practice in industry. It provides employees with a more engaging work environment, resulting in far less monotonous and repetitive tasks [14].

1.2. Definitions and Contextualisation for Core Terms

Section 1.1 has motivated the introduction of human-robot teaming into the manufacturing industry. Researchers with various research backgrounds explore the interplay of humans and robots, each using distinct terminology (e.g. 'task' is used in different interpretations). Consequently, it is essential to provide precise definitions for key terms to distinguish which research directions are covered in this thesis. The following sections thus introduce technical terms from three domains that are essential for our work: from the domain of industrial assembly, from the domain of computer vision, and from the domain of human-robot collaboration.

1.2.1. Terms from the Domain of Industrial Assembly

Each industrial assembly process has an application-specific *goal*—an objective that is to be completed. A goal might, for example, be the complete assembly of a certain product (e.g. a car engine, a mobile phone). This goal is achieved within a certain *environment*, which includes the robot, the workcell, sensors, tools, and the workpieces.

An *agent* is every entity that perceives the environment, reasons about it, and can execute actions within the environment. Both humans and robots are agents. We understand an *action* to be a fundamental manipulation of the environment that is atomic, quickly conducted (e.g. in seconds) and—ideally, at least—fail-safe. For instance, placing a block on a table constitutes an action. Fail-safe means that we do not model erroneous action execution, e.g. placing a block with an offset of $2\,\mathrm{cm}$ to the target location. Since the assumption of fail-safe actions does not hold in practice, this thesis also discusses the detection of action failures, but only outside the context of modelling.

A *task* is a set of actions and their dependencies which are necessary to achieve the goal. All agents collaborate to achieve the task. Section 5.3 introduces a formal model of a task. Examples of tasks include the assembly of a product or the disassembly of a product into its constituent parts. While the goal defines the outcome of the task, the task describes the steps necessary to achieve the goal and the dependencies among the steps. A task encodes multiple ways to achieve the goal, e.g. in which order an agent fixes a row of screws. During task execution, each agent exhibits a specific behaviour. A *behaviour* is a set of rules that determine which actions an agent executes next, e.g. tightening screws from left to right. If the agent is a robot, its behaviour is also referred to as policy. A *policy* is the mapping from the current progress in a task to the next action the robot will execute. Our assumptions include an industrial-like structured environment where all agents are aware of tasks and actions, and object positions remain fixed.

1.2.2. Terms from the Domain of Computer Vision

In addition to the industrial assembly domain, this thesis employs terminology from computer vision. This thesis envisions the following configuration: a top-mounted camera for tracking the progress of the task, a human and a robot facing each other, and a task description that is known to both (Figure 1.2). While this specific configuration is not an absolute prerequisite for all software components, it significantly influences their design and the underlying assumptions. Most of the algorithms presented in this thesis operate on external computers connected to the robot. Consequently, we collectively refer to the robot, these computers, and the camera as the *system*. The camera is a time-of-flight RGB-

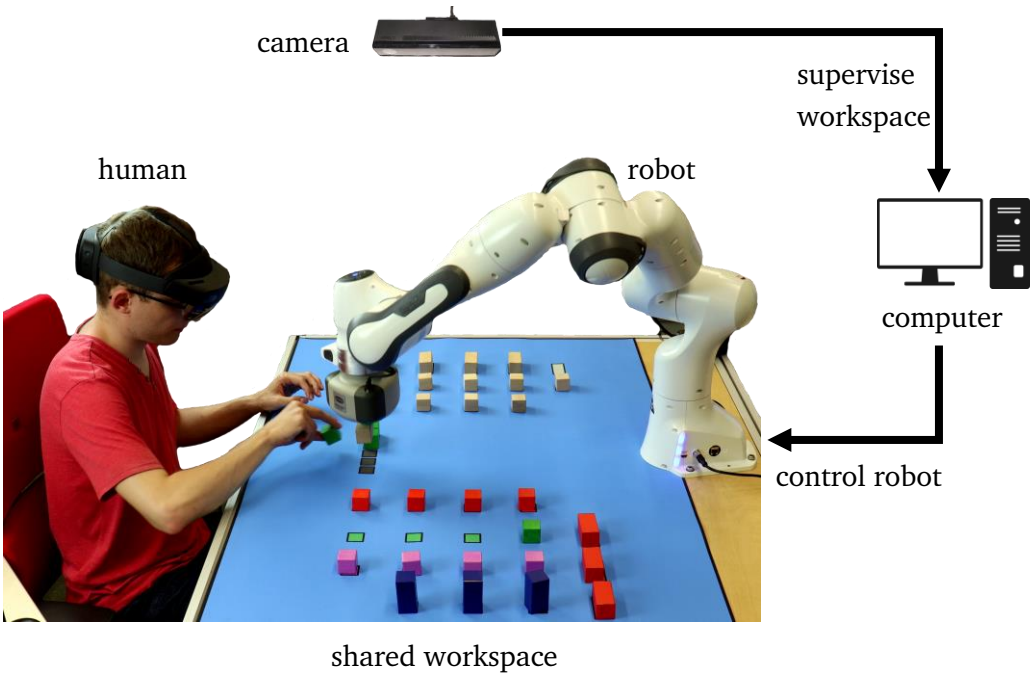


Figure 1.2.: Schematic drawing of the envisioned setup. The computer on the right runs all the algorithms described throughout this thesis. The coloured blocks are a mock-up for assembly components (e.g. screws, casings, system boards)

D camera [15]. It captures the conventional colour channels, and it measures per-pixel distances to environment objects by time-of-flight of an infrared pattern. The camera provides data in the form of video *frames*, which arrive with a fixed frame rate. As a consequence of the positioning of the camera, certain objects are not visible due to the presence of one of the agents between the object and the camera. This phenomenon is referred to as *occlusion*. Section 5.1 elaborates on why this setup was chosen and which alternatives exist.

1.2.3. Terms from the Domain of Human-Robot Collaboration

Following up on the technical terms from industrial assembly and computer vision, we conclude with essential technical terms from human-robot collaboration. The fundamental term of our work is the term of *team*:

> *A team is described as a relatively small group of partners with each complementary skills who are committed to a single objective, result target, as well as plan and hold each other accountable for it. The same may be said about human-robot teams, in which both humans and machines are dedicated to working together to accomplish a common goal. [10]*

The above definition considers humans and robots as superficially exchangeable. A deeper look, however, exposes some evident differences. Notably, humans—in contrast to robots—are naturally uncontrollable agents and prefer their actions not to be imposed. So, their behaviour can only be estimated and emulated. This thesis assumes that humans are cooperative, rational, and congruent. However, their involvement, computational capabilities, and tolerance regarding the shared task can vary [16]. As part of the assumption of error-free task execution, each human must follow some premises: the human adheres to the task plan, tries to avoid conflicts (e.g. by not grabbing the same object shortly before the robot does), and executes the task in a reasonable, non-chaotic way. Moreover, we assume that the human has an in-depth understanding of the task such that he or she is capable of fixing errors—if these occur. Instructing the human to execute the task is not a focus of this thesis.

Given a human-robot team, the members of the team need to coordinate their actions. The coordination mechanisms for a human and a robot in a shared working environment can be categorised into coexistence, interaction, cooperation, and collaboration [14, 17]. *Coexistence* means that there is no physical barrier between the human and the robot while both work on different tasks. *Interaction* is distinguished from cooperation

and collaboration by the sequential nature of execution. There is no overlap between human actions and robot actions in the temporal domain. *Collaboration*—in contrast to *cooperation*—involves direct physical contact between the human and the robot or the simultaneous exertion of force onto an object by both team members.

In the following, we use the umbrella term *human-robot teaming* to refer to human-robot coordination ranging from interaction to collaboration. Key characteristics of teaming are that humans and robots work in a shared workspace on a common task. Actions can be executed in parallel or sequentially. Direct contact or the simultaneous exertion of force onto an object is possible, but not the focus of this work.

An orthogonal categorisation to coordination mechanisms is agent autonomy. *Agent autonomy* 'expresses how much of robot action is directly determined by human agents, and vice versa' [14]. It is often discussed in the context of a leader-follower relationship. The *leader* is the most influential agent of a team—for instance, the leader may carry out major actions by him- or herself, and the leader may decide over the actions of the follower. The *follower* only executes supportive actions such as applying fixtures, or handing over tools or workpieces. Its actions are largely determined by the preceding and subsequent actions of the leader. The working speed depends on the leader. Agent autonomy can, to some extent, be measured by neglect tolerance. *Neglect tolerance* quantifies the timespan a system can work without human intervention [18]. High neglect tolerance indicates high autonomy of the robot. To increase neglect tolerance, the roles of leader and follower can be reassigned during task execution, referred to as *shared autonomy*.

Another perspective on autonomy is the responsibility for *action allocation*, the assignment of actions to specific team members for subsequent execution. The literature distinguishes static and dynamic teaming [19]. In *static teaming*, all action allocations are made before task execution. In contrast, *dynamic teaming* refers to 'situation-dependent co-working' [19]. A mixture of both is *semi-dynamic teaming* where agents make decisions within their pre-determined role.

Introducing these teaming approaches comes with increased complexity when conducting the work. Approaches must therefore prove worthy to justify their introduction. Proposed benefits include improved productivity, more flexible production lines, and improved job quality [19]. To produce meaningful scientific insights on the proposed benefits, approaches for human-robot teaming must be benchmarked with appropriate evaluation strategies, benchmark tasks, and evaluation metrics in line with the overarching goals, as depicted in Figure 1.3.

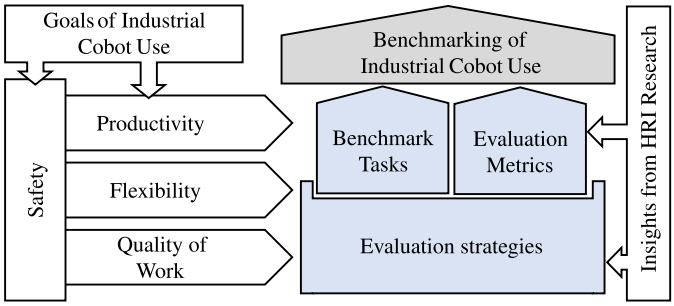


Figure 1.3.: Key steps of benchmarking industrial cobot use [19]: Benchmarks require appropriate tasks, metrics and evaluation strategies. Suitable tasks, metrics and strategies must, on the one hand, consider the goals of industrial cobot use and may, on the other hand, be inspired by insights from HRI research.

Productivity relates to the time or cost of manufacturing a single product or parts thereof. Productivity metrics are further subdivided into efficiency and teaming fluency. *Teaming fluency* subsumes metrics which explicitly measure the load factor of each agent or production resource. *Efficiency*, in contrast, focuses on the resources invested to produce a single product. *Flexibility* refers to the number of variants the human-robot team can adapt to during task execution. Those variants can either refer to the time it takes to switch to another task, referred to as *task flexibility*, or the distribution of workload, referred to as *teaming flexibility*. Both productivity- and flexibility-related metrics have the fact in common that they only rely on objectively quantifiable parameters. Quality of work, in contrast, wants to capture the subjective experience of workers. Quality of work is subdivided into physical and cognitive ergonomics. *Physical ergonomics* assesses loads impacting the human body to prevent disorders of muscles, nerves, and joints. By contrast, *cognitive ergonomics* aims at mental health and subjective comfort. For physical ergonomics, a broad range of assessment tools and workspace guidelines exist.

Measured concepts for cognitive ergonomics have largely been inspired by human factors analysis and then extended to teamwork scenarios. The most prominent concepts are cognitive workload, affect, psychological safety, satisfaction, subjective performance, acceptance, the robot's perceived personality, interaction quality, and trust [19]. Interaction quality subsumes the term fluency. *(General) fluency* is 'the subjective experience of ease or difficulty associated with completing a mental task' [20]. Extended to teaming, this means: '*Collaborative fluency* is the coordinated meshing of joint activities between members of a well-synchronised team' [21]. Characteristics of collaborative fluency are that actions are ordered and timed in such a way that the interaction feels comfortable and natural. Collaborative fluency has a positive impact on team performance [22, 23]

Goal	Evaluation Metrics	Description
Productivity	Concurrent Activity	Fraction of time both agents work
	Human/Robot Idle Times	Complement of concurrent activity
Flexibility	Intervention Rate	fraction of time during which a human controls the robot
	Neglect Tolerance	fraction of time the system can work on its own
Job Quality	NASA Task Load Index (NASA-TLX)	Questionnaire on mental workload
	Fluency	Questionnaire on teaming traits

Table 1.1.: Selection of metrics presented in [19] relevant for this thesis.

and user satisfaction [24]. Throughout this thesis, we abbreviate collaborative fluency as *fluency* since we do not discuss other types of fluency. Excessive cognitive workload causes stress, anxiety, fatigue, and an increase in error rate. All of them negatively affect the worker's health and performance [25]. Hence, cognitive ergonomics has a long-term positive impact on product quality and productivity.

As a benchmark metric, cognitive ergonomics forms a major aspect of this thesis. Supporting flexible dynamic teaming and flexible task allocation are secondary aspects throughout this thesis. They do not give rise to benchmark metrics but instead serve as core design principles. Productivity plays a minor role. The goal of this thesis is not to implement a productive system but to evaluate concepts of flexibility for their cognitive ergonomics.

Table 1.1 gives an overview of **evaluation metrics** relevant to this thesis. By design, this thesis aims at intervention rates close to $0\,\%$ and neglect tolerance close to $100\,\%$. As a consequence, pick-and-place tasks with a scalable degree of complexity and interdependence are used. This has effects on the required perception capabilities of the system, as explained in Section 5.2. A thorough discussion of benchmark tasks and evaluation metrics is presented in Section 7.1.

According to [19], the main **evaluation strategies** are research demonstrators, human-participant studies, and virtual commissioning. *Research demonstrators* are physical systems showcasing the technical feasibility of the approach. They can serve as a common ground for structured interdisciplinary dialogue. They are, however, very limited in terms of metrics that can be evaluated on them without involving human subjects. The next step is thus conducting *human-participant studies* (referred to as *user studies*) with research demonstrators, where humans interact with the system in a controlled environment on

a predefined task. They offer the broadest range of gatherable measures. However, they require the largest effort to prepare and conduct studies. Virtual commissioning overcomes many of the technical challenges by replacing the physical system with a real-time simulation. Studies with human subjects are, in this context, referred to as *immersed human-in-the-loop virtual commissioning* (HIL VC). HIL VC uses Augmented Reality or virtual test beds to simulate the robot system. Interacting is then achieved by keyboard or controller inputs. However, the reduction in complexity comes at the cost of simplified physical effects and the risk of motion sickness.

The goal of this thesis is to contribute towards improving cognitive ergonomics. Studies with human subjects are therefore necessary. Human studies can be carried out with a physical and a virtual system. A physical system—in contrast to a virtual one—exhibits two special effects of close cooperation: On the one hand, a real robot moving close to a human subject has a certain psychological effect and, on the other hand, physical tasks are susceptible to small disturbances that have to be corrected by the human. To accurately capture both effects, this thesis pursues a physical cooperation setup.

1.3. Problem Formulation and Research Questions

To sum up, static teaming allows for pre-computing and optimising throughput times if the execution times of each action are known in advance. This creates plannable cycle times for workstations where the human follows the cycle of the machine. However, static teaming is not an end-all solution: The human's situational awareness and capabilities to detect errors suffer under static teaming [24]. Neglect tolerance and therefore flexibility of the overall system are low as well. Interviews of shop-floor workers [26] and lab experiments [24] show that humans prefer to have some control over the robot system rather than completely following its schedule. Yet, explicitly instructing the robot alongside doing their own part of the task also overburdens human workers [24]. The sweet spot hence lies in the middle, where the robot proactively executes actions and adapts to the human. Proactive task execution by the robot, in turn, contributes to shared autonomy.

This thesis proposes, implements, and evaluates mechanisms that allow the robot to adapt to the human. *The central working hypothesis of this thesis is that subjective fluency benefits from behavioural adaptation based on action prediction*. Focusing on the teaming aspect, the user's specification of the task exceeds our scope. The reader is referred to related work on how users can specify tasks for human-robot teaming [17, 27].

In the course of this thesis, we approach our central working hypothesis by means of six research questions. We derive these questions from advanced key capabilities a robot might require to adapt to a human co-worker. Research on human teaming subsumes the process we consider here under the broader term of joint action. *Joint action* is the process where multiple agents engage in social interaction to coordinate their actions towards changing the environment.

Despite the definition of joint action extending far beyond coordination for assembly tasks, we can still draw some insights from the research on joint action and derive required robot capabilities. Prior research has identified five main mechanisms necessary to perform joint action [28]: (1) action coordination, (2) perception of agency, (3) joint attention, (4) action observation, and (5) task-sharing. Let us now discuss the relevance of each of these mechanisms for this thesis and its research questions, if applicable.

Action coordination refers to the allocation of actions to agents and the process of negotiating that allocation. We address action coordination as part of the robot's action planning for the prototype system. *Perception of agency* is tightly coupled with action coordination. The term refers to 'the ability to distinguish among actions of different actors and their effects' [29]. This aspect of joint action becomes relevant when one of multiple concurrent actions has to be determined as the cause for a certain effect. In the context of this thesis, however, perception of agency is negligible because we assume our actions to be atomic and therefore, our actions can never be concurrent under the premise of the same effect.

Joint attention denotes the communication channels that team partners use. Communication channels can further be categorised into speech (e.g. directed or undirected verbal instructions), gesture (e.g. by hand, head, or face), and action (e.g. by order or timing thereof) [30]. In the context of human-robot teaming for industrial assembly, communication channels incite three observations: First, all three channels are unreliable to a certain extent, may that be due to complex semantic relationships, due to ambiguities or simply due to noise. Second, speech has proven to be particularly unreliable [31]. Third, current robots—as opposed to humans—additionally have limited cognitive capabilities, no matter the particular communication channel.

Combining the above observations forms the rationale behind our first research question: We assume a worst-case environment where speech is not possible, and we assume human-like cognitive capabilities of all teaming agents. Under these assumptions, we want to evaluate the effectiveness of fallback communication channels (i.e. in the absence of speech). To simulate the contribution a future, human-like robot could hope to

achieve in this scenario, we substitute the robot with an actual human. This gives rise to our first research question:

Q1 How effective is teamwork in the absence of speech? Which other channels of communication do humans use?

Action observation is the mechanism by which one agent perceives and remembers the actions of other agents. Action observation is required to understand communication intents by action, but also to keep track of what parts of the task have been completed. For action observation, the robot requires the ability to perceive and reason about the progress of the task. The envisioned setup allows the human to assign actions to themselves without explicit communication with the robot. Consequently, there is no predetermined schedule, and, in turn, observing a single action is insufficient for the robot to derive the overall progress. Consider an example where the human has to tighten a row of screws, and the schedule forces the human to do it from left to right. If the robot observes the tightening of the right screw, the robot can conclude that all screws have been tightened. Our setup allows the tightening of the screws in arbitrary order. Thus the robot must keep track of each screw individually.

Apart from the uncertain order of actions, another source of uncertainty stems from the perception setup. Observing the workspace with RGB-D cameras yields the benefits that the equipment is non-invasive and easy to install. The downside is that the robot and the human may occlude parts of the workspace. On top of this, shadows and varying light conditions prevent the accurate detection and classification of every object in the workspace at every point in time. The robot, in turn, may miss actions performed by the human. The following example illustrates the problem: The robot reaches for a faraway object in a way that makes the robot occlude a large part of the workspace. In the meantime, the human picks an occluded object from below the robot arm and puts the object onto the assembly. Due to the occlusion, the robot could not observe the pick action but only the place action. If multiple instances of the same object had existed below the robot arm, any of them could have been picked.

Due to the arbitrary ordering of actions and due to perception uncertainties, our system cannot always maintain a single ground-truth task state. At the same time, our system must still satisfy soft real-time requirements (e.g. to achieve fluency), so computationally expensive alternatives (e.g. tracking all combinatorially possible states) are beyond question. Our second research question thus explores computationally efficient alternatives to state tracking that consider both flexible teaming and missing information from the sensor setup:

Q2 How can the task state be efficiently updated under observation uncertainties in a flexible task model?

Task-sharing refers to building mental models of the task and the other agents. *Mental models* are small-scale models of reality that are used to predict events [32]. In the context of teamwork, mental models are extended to *shared mental models* (SMMs). SMMs 'are thought to provide team members with a common understanding of who is responsible for what task and what the information requirements are. In turn, this allows them to anticipate one another's needs so that they can work in sync' [33]. SMMs therefore represent synchronised mental models with common expectations.

SMMs are categorised into first- and second-order models. While first-order models only consider direct intentions of a team partner, second-order models capture how team members reason about others. This work only considers first-order models.

First-order SMMs already achieve desirable traits, such as fluent behaviour, adaptability, trust building, effective communication, and explainability [34]. It seems obvious that SMMs can lend these benefits to human-robot teams as well: In the context of human-robot teaming, humans tend to attribute team membership to robots, and if humans work with robots on the same assembly (rather than on separate ones), they perceive the robot as even more of a team member [35]. Conversely, not only should the human maintain an SMM of the robot, but the robot should maintain an SMM of the human as well.

Establishing an SMM of a human co-worker is not trivial: Humans are capable of a broad behavioural inventory—both in terms of executing actions and in terms of predicting others' next actions [36]. For that, they use a wide variety of cues ranging from planning constraints over social conventions to explicit transition probabilities for task states. Adults have an accurate representation of these transition probabilities for everyday activities. They are also proficient in learning artificial ones [36].

As a first step towards SMMs for human-robot teaming, we examine the best-case SMM that a future, human-like robot could hope to form based on its limited perception capabilities. We do so by examining the SMM of an actual human with artificially limited sensory input instead of a human-like robot. To assess the quality of said SMM, we draw on the accuracy of the SMM prediction for the next action of other agents. In the words of our third research question:

Q3 To what extent can human observers with limited sensory input predict action sequences of other humans working on an assembly task?

The answer to the above research question serves as a reference for what a future, human-like robot could envision to achieve. From a practical point of view, such as for an actual human-robot collaborative application, there is a challenge associated with this vision: An SMM with, at best, human-like capabilities must be replicated in the form of an algorithmic solution. The replicated model must nevertheless be able to encode and learn the behaviour rules specific to a human teaming partner to gain a better understanding of his workflow [37]. Any capabilities and limits of the model, in turn, shape the capabilities and limits of the robot and of the human-robot team. For example, a minimum requirement for a supportive robot is that the robot—as controlled by the model—does not hinder the human's task execution. In the context of our framework, this translates to not starting the same action as the human.

The algorithmic realisation of a human-like SMM and an investigation into its capabilities and limits drive our fourth research question:

Q4 To what extent can action sequences of humans be algorithmically predicted from previous behaviour?

The answers to research questions Q2 and Q4 introduce novel concepts that allow the robot to perceive and reason about flexible task execution. It is therefore necessary to evaluate their effect on the outcomes of joint action. We do so in our fifth research question. As motivated above, the focus is on fluency and cognitive ergonomics:

Q5 How does action prediction impact cognitive ergonomics in an industrial-like human-robot teaming scenario? What is the specific effect on fluency?

The answer to research question Q5 involves a variety of metrics from the literature. In this regard, Section 1.2.3 has already motivated fluency as one of these metrics. Fluency, thereby, is a relatively new concept within the overall scope of quality of work. As the authors of the concept note, the questionnaire for evaluating the concept needs some fine-tuning. The concept's practical relevance and benefit in addition to other metrics remain an open question. This thesis therefore seeks to contribute further insights into fluency metrics in the context of human-robot teaming by answering the sixth research question:

Q6 How are objective and subjective metrics with respect to productivity and fluency related?

Finally, we return to the central working hypothesis of this thesis: Our findings for research questions Q5 and Q6 show whether subjective fluency benefits from behavioural adaptation based on action prediction. Research questions Q1 to Q4 provide the necessary technical and methodological foundations.

1.4. Conceptual Overview

The thesis is structured as follows. Chapter 2 reviews general, related literature for human-robot teaming. It focuses on three overarching aspects of this thesis, namely how dynamic teaming has been achieved, how shared mental models have been addressed and how the fluency of the human-robot teams has been investigated. Their purpose is to give a general impression of the main research direction and how our approach differs from it. The literature presented there has in common that approaches are evaluated by means of user studies for human-robot teaming or interaction.

Chapters 3 to 6 present and evaluate the main components of the system, addressing research questions Q1 to Q4. Each chapter starts with a literature review related specifically to this chapter. Each review addresses the specific challenges of that component and presents advancements from beyond our application domain.

Chapter 3 establishes a reference for what the robotic system with human-like cognition but technically restricted perception could achieve. Two user studies are employed to investigate research questions Q1 and Q3. The first user study addresses Q1 and, as such, the effectiveness of cooperative assembly without verbal communication (Section 3.1). The second user study addresses Q3 and, as such, the capabilities to infer a shared mental model (Section 3.2). Insights from these studies drive the design of the following components of the system.

Chapter 4 is preparatory work for research question Q2. It addresses hand tracking as a measure to reduce the large set of actions the human can potentially execute. The chapter presents an approach that combines data from multiple frameworks to estimate hand poses. A pipeline is introduced to achieve hand tracking (Section 4.2). The pipeline combines classic approaches for skin segmentation and hand region tracking using bounding boxes (Section 4.3) with neural networks for pose estimation (Section 4.4). Information is then fused by iterative pose refinement (Section 4.5). Finally, the pipeline is evaluated and augmented with further data from the HoloLens 2 to increase robustness (Section 4.6). The output is then a continuous and smooth hand trajectory.

Chapter 5 addresses research question Q2. The chapter describes the pipeline from the sensory data to the detected action sequences. Hand-crafted object detection and

classification detectors are the starting point (Section 5.2). Additionally, these detectors provide information about occlusion. The task model is formalised in terms of a coloured Petri net (Section 5.3). The task model allows a flexible task execution for both human and robot. However, this requires a new approach to tracking the task state (Section 5.4). The new approach combines the data from object classification and hand tracking. Selecting the next action for the robot is non-trivial in such a flexible task model. Thus, an algorithm to filter possible next actions is presented (Section 5.5). The algorithm is necessary to control the robot in Section 5.7 and Section 7.3. Finally, the task progress tracking is tested in a simulation environment and compared against a baseline approach (Section 5.7).

Chapter 6 addresses research question Q4. The chapter uses the recorded action sequences to learn the mental model of the human. Notably, the resulting model predicts the next human action given the previous actions. This serves as a link to Section 3.2, where the human capabilities of action prediction are investigated. We first introduce a formalisation of general assembly rules (Section 6.2) as a baseline approach. Next, actions are encoded into a custom feature space that encodes the spatial information (Section 6.3). The encoded actions are forwarded into a deep neural network (Section 6.4). Inputs to the neural network are the previous actions and a candidate for the next action. The neural network then outputs how likely it is for the candidate to be the next action. Section 6.5 describes how the network is integrated into the overall software and how it is trained.

Chapter 7 integrates the evaluation of the action prediction component into the overall evaluation of the software. Section 7.2 describes the decision-making and action execution of the robot. Section 7.1 addresses action coordination for the robot. The action prediction from Chapter 6 is extended towards intention prediction through repeated invocation. The outcomes then determine the robot's action selection. Section 7.3 explains the setup for the user study that evaluated the demonstrator for the system. The study is then evaluated in terms of fluency (Section 7.4.1), the correlation of fluency and productivity metrics (Section 7.4.2), as well as the performance of intention prediction (Section 7.5). These subsections provide the quantitative answers to research questions Q4 to Q6.

Chapter 8 concludes this thesis. In this chapter, we summarise the main findings of this work, answer the research questions and point out future research directions.

CHAPTER 2

Related Work

The previous chapter motivates the need for dynamic human-robot teaming. It highlights several relevant research questions ranging from perception of the task state to coordinating human-robot teams. Given that the research field of human-robot teaming is well-established already, the current chapter reviews prior work relevant to the broader context of this thesis.

We discuss related work in three categories—from the implementation of dynamic human-robot teaming for assembly tasks (Section 2.1), over the formalisation and implementation of shared mental models (Section 2.2), to the investigation and analysis of fluency (Section 2.3). In all three categories, we focus on related work that conducted human studies. As motivated in Section 1.2.3, our focus accounts for the fact that other approaches are too limited in measures to derive meaningful real-world insights.

2.1. Dynamic Human-Robot Teaming Approaches

Many approaches in literature assign a dedicated assistant role to the robot (e.g. [38, 39, 40, 41, 42, 43, 44], see Table 2.1). The robot's only task is to fetch the right parts at the

Characteristics	Teaming Mode	Autonomy	Con. & NT	Examples
Robot fetches parts	static	human-led	medium Con., low NT	[38, 39, 40, 41, 42, 43, 44]
Turn-taking	dynamic	shared	low	[45, 46, 47, 48]
Assembly with strict roles	static	shared	high Con., low NT	[49, 50]
Adaptive scheduling	semi-dynamic	robot-led	medium	[51, 52]
Mutual adaptation	dynamic	shared	potentially high	[53, 17]

Table 2.1.: Categorisation of human-robot teaming approaches ordered by increasing level of autonomy, concurrency (Con.), and neglect tolerance (NT). Only a few approaches achieve high levels in all three aspects.

right time. This has two consequences: First, the assembly process totally depends on the human partner—in other words, the robot has a low neglect tolerance and cannot be productive without the human partner. Second, due to the nature of hand-overs, the shared workspace is small. Spatial conflicts rarely occur, and task sharing is not possible.

In general, a dedicated assistant role for the robot prevents extensive, dynamic teaming. Minor aspects of dynamic teaming can still be applied under this role assignment: On the one hand, in the case of fixed execution orders, the approaches aim for optimal timing (e.g. [44, 40, 41]). On the other hand, if the human has some autonomy over action selection, predicting the next action and offering the correct support becomes relevant (e.g. [38, 39, 41]).

With more capabilities, the robot can make a larger contribution to the task. This can take the form of human-robot interaction where human and robot take turns (e.g. [45, 46, 47, 48]) or they are assigned fixed, pre-determined roles and only execute their type of actions (e.g. [49, 50]). Still, both variants exhibit low neglect tolerance and adaptability. More flexibility is gained when there is a subset of actions that both agents can execute. The scheduling procedure can then take the varying execution times of the human into account to minimise overall execution time [51] or fatigue [52]. Both approaches exhibit a higher neglect tolerance as the robot can work on its own for some time. Yet, human autonomy is still low as all scheduling decisions are made by the robot. Lamon et al. [54] increase human autonomy. Users can decline subtasks, which are then reassigned to the robot. Schmidbauer et al. [26] let users specify their allocation preferences before task

execution. Both approaches allow the user control over the scheduling but come at the cost of increased configuration overhead and limited re-scheduling capabilities during execution.

Truly dynamic cooperation and autonomy during task execution can only be found in Baraglia et al. [53] and Riedelbauch [17]. Hence, a more detailed discussion of both works is appropriate. In either work, actions are assigned on the fly. Likewise, in either work, the human and the robot individually track the current state of the task and pick appropriate next actions.

As its unique contribution, Baraglia et al. [53] demonstrated a scenario where human and robot have to work together to prepare a table setup. Due to reachability constraints, none of them can execute the task alone. Task planning is achieved by leveraging Dynamic Bayesian Networks and spatial reasoning. Action selection is bound to perception and proximity. Actions are only executed on objects that have been observed robustly and consistently in previous frames; actions on closer objects are preferred.

The experiments of Baraglia et al. [53] show that concurrent motion can be achieved. However, concurrency does not exceed $4\,\%$ of the overall execution time. The reasons are the low number of objects making parallelisation difficult and the human being magnitudes of orders faster at pick-and-place actions than the robot. To fill the gap, the authors included logging activities. After each completed action, the user has to input the completed action into a computer. Despite the capabilities of the approach, the participants in the study often interacted or coexisted rather than cooperated with the robot.

In contrast to Baraglia et al. [53], the work by Riedelbauch [17] presents a setup with highly parallel execution. Both agents dynamically plan what to do next. A precedence graph is used as a task model. Tracking of the task state is achieved in terms of pre- and postconditions for each action. That way, the start and completion of each action can be tracked. The uniqueness of the approach lies in its sensor placement. There is a single camera attached to the end of the robot that is close to the gripper. This simplifies sensor setup but comes at the cost of a small viewport restricted to the area right under the robot gripper.

The experiments of Riedelbauch [17] show that both agents work truly in parallel and assign actions on the fly. However, the limited view of the robot prevents it from perceiving most of the user's actions. It is for this reason that Riedelbauch [17] can not perceive actions globally. Instead, actions are only recognised locally and possibly with a delay—notably, once the camera observes any resulting changes in the environment.

In turn, the robot may schedule actions that the human has already executed, and the robot needs extra time to check the task state by moving to an appropriate point of view.

Collectively, all preceding considerations show a research gap: There is no dynamic cooperation framework as in Riedelbauch [17] with a global perception of human actions.

2.2. Modelling Shared Mental Models

Many approaches in the field of dynamic human-robot teaming make use of a *Markov Decision Process* (MDP) for SMMs. MDP is a discrete-time stochastic control process. It models a controllable agent (the robot) and an uncontrollable, but stochastically predictable environment (the human). An MDP is a 4-tuple (S, A, P_a, R_a) with the following semantics:

- $S = \{s_1, s_2, ..., s_{|S|}\}$ the set of all possible task states,
- $A = \{a_1, a_2, ..., a_{|A|}\}$ the set of all actions executable by the robot,
- $P_a : S \times S \rightarrow [0, 1]$, $P_a(s_i, s_j) = P(s_j|s_i, a)$ the transition probability that robot action $a \in A$ in task state $s_i \in S$ will lead to task state $s_j \in S$, where the actual probability value incorporates the stochastic action selection process of the human, and
- $R_a : S \times S \rightarrow \mathbb{R}$, $R_a(s_i, s_j)$ the immediate reward after transitioning from task state $s_i \in S$ to task state $s_j \in S$ due to robot action $a \in A$.

The following example illustrates how transition probabilities $P_a(s_i, s_j)$ are constructed. Assume we are in a state s_4 and the robot takes action a_7 to reach state s_9. After that, the human will always do some unspecified action of his or her own, which always makes the task progress to state s_2. Then, $P_{a_7}(s_4, s_2) = 1$ and $P_{a_7}(s_4, s_k) = 0$ for all $s_k \neq s_2$.

The transition probabilities $P_a(s_i, s_j)$ combined with immediate rewards $R_a(s_i, s_j)$ form the expected future rewards. Maximising over expected future rewards yields the optimal action policy of the robot.

Nikolaidis et al. [55] learn transition probabilities from a training phase in a simulated environment. Human and robot then switch roles in the training, such that the human can execute the preferred robot response action. From the switched task execution, the robot can compute the rewards. The state space of the task consisted of only 27 states—which is small enough to sample transition probabilities from demonstrations.

Nemlekar et al. [56] take one step towards a generalisation of the reward function. Instead of individual rewards per state-action pair, they learn weights for a linear function. Input to the linear function are features (e.g. physical effort rating or keeping the same part) derived from the robot action, previous action and task state. Weights are learned from a training session where the user needs to explicitly specify subjective ratings for each action. Since preferences are already encoded in the reward function, the transition probabilities are uniformly distributed.

Some approaches take a step further from modelling or predicting the next action towards higher-level concepts such as trust, fatigue, an information processing model, or the task goal assumed by the user. The approaches use extensions of MDPs towards hidden states (e.g. [47, 48, 57]). In case of extended MDPs, the model is no longer in a single state, but has a probability distribution over states, called the *belief state*. The belief state resembles the user's level of trust [57], the human's model of the task state [48], or of the common goal [47].

In all of the above approaches, MDPs are mostly used for turn-taking, which is in line with their original design. This limits their applicability to interaction scenarios. Furthermore, whether MDPs are even able to cope with a more complex task and the incurring curse of dimensionality in the task space remains unclear. It is quite obvious, at least, that a naive approach to constructing a stochastic model from training samples for all state-action pairs becomes infeasible if the task space is sufficiently large.

2.3. Investigation of Fluency in Human-Robot Teaming

Fluency is a major indicator of the benefits of human-robot teaming. For example, in prior work, fluency has been used to evaluate the effects of scheduling policies (e.g. [51, 59, 60]), proactive robot behaviour (e.g. [56, 53]) and cross-training (e.g. [55]).

To gain a scientific understanding of fluency, fluency must be measured. While the measure of fluency is still at an early stage, its investigation has been gaining momentum in the robotics research community (e.g. [60, 56, 51, 64, 61]). The research community has to date proposed several categories of metrics to capture the construct of fluency [19]: (i) self-reports on fluency (e.g. [21]), (ii) productivity-related proxy metrics (e.g. [21, 18]), and (iii) related outcome variables (e.g. user satisfaction [65]).

In terms of self-reports, questions are often tailored to one dedicated study setup and are not readily reusable for other studies (e.g. [66, 53, 58, 62]). But from a variety of these questions, a comprehensive, re-usable list of select questions has been compiled [21]. The questions are rated on a 7-point Likert scale from 'strongly disagree' to 'strongly

	Metrics for fluency			
	Self-reports	Productivity-related	Outcome variables	Dependencies evaluated
Baraglia et al. [53]	●	●	●	◐
Chao and Thomaz [58]	●	●	●	○
Chao and Thomaz [59]	●	●	●	◐
Favier and Alami [60]	●	○	●	○
Gervasi et al. [61]	●	●	●	◐
Lasota and Shah [62]	◐	●	●	○
Nemlekar et al. [56]	●	●	◐	○
Nikolaidis et al. [55]	○	●	●	○
Petzoldt et al. [51]	●	●	●	○
Shah et al. [63]	●	●	●	○
Tsitos and Dagioglou [64]	●	●	●	○

Table 2.2.: Categories of metrics for fluency used to evaluate dynamic human-robot teaming (○ = not used, ◐ = few or loosely related questions, ● = thoroughly covered) and evaluation of dependencies among these categories (○ = not evaluated, ◐ = some evaluated).

agree'. Ratings are assigned a numerical value and averaged to obtain a value for the scale of fluency. The proposed scale of fluency is validated against three reference scenarios that are supposed to induce low, middle and high fluency. The questionnaires have been used to evaluate human-robot teaming (e.g. by [63, 60, 64]).

Productivity-related proxy metrics include completion time, human and robot idle times, as well as concurrent action times (e.g. [55]). They rely on the assumption that fluency manifests in reduced waiting times. Since researchers are aware that his assumption does not necessarily hold, many studies use a combination of self-reports and proxy metrics (e.g. [53, 59, 51, 56, 61]).

Related outcome variables are other metrics from cognitive ergonomics that might be influenced by fluency. These are, for instance, interaction quality (e.g. measured by e.g. PeRDITA [67]), user satisfaction (e.g. measured by SUS [65]), and cognitive workload (e.g. measured by NASA-TLX [68]). Many of the above-mentioned approaches (e.g. [51, 59, 60, 53, 55]) include questions in their self-reports that are targeted towards other dimensions of cognitive ergonomics rather than fluency.

Despite collecting metrics from several of the above-mentioned categories, metrics have mostly been evaluated in isolation from one another (Table 2.2). The authors of [61], for example, use isolated metrics to show that the robot's movement speed and

human's control—i.e. the human presses a button to start the robot action—influence fluency. More precisely, they rely on interaction quality, physiological responses and affects as metrics, but they do not consider any correlations between these metrics.

A correlation of metrics may offer additional scientific insights. In particular, non-correlating metrics indicate that these metrics capture different underlying concepts. Baraglia et al. [53] and Chao and Thomaz [59] are the only works that consider confounding factors in their metrics. We therefore discuss both methods in detail.

Baraglia et al. [53] investigated three robot behaviours. The behaviours differed in whether the robot acted by itself or participants had to ask for help. The study evaluated overall completion time and interaction quality (as a combined metric including fluency) for each of the three behaviours. The authors found that interaction quality is not directly related to completion time and efficiency. As noted in Section 2.1, the simultaneous movement of the partners was very low and was more comparable to an interaction scenario.

In a different setup, Chao and Thomaz [59] directly measured the elapsed time between human and robot action (referred to as *response delay*). They found that reduced response delay correlates with fluency and responsiveness.

Both Baraglia et al. [53] and Chao and Thomaz [59] show that fluency is linked to response delay, but does not correlate with completion time and efficiency. Their studies however, primarily depict interaction scenarios without concurrency. In such scenarios, the human must wait for the robot. It thus remains worthwhile to investigate whether the findings still apply in a dynamic teaming approach with high concurrency.

Established measures from cognitive ergonomics were not collected in the studies by Baraglia et al. [53] and Chao and Thomaz [59]. Yet, interaction quality and cognitive ergonomics are overarching concepts for fluency. To what extent these overarching concepts can discern differences stemming from fluency remains worthwhile to investigate, as well.

2.4. Conclusions and Discussion

The above literature review depicts approaches from dynamic human-robot teaming with user studies. Key insights from the related work are: Most approaches are restricted to interaction scenarios or assign fixed roles to human and robot (Table 2.3). As such, the approaches have a low neglect tolerance. If the human does not perform an action in time, the robot cannot be productive either. Some of them allow dynamic teaming or shared autonomy, but not both at the same time. Higher neglect tolerance is only encountered by

Maintain a SMM of the Human?	Teaming Mode	
	Interaction or Fixed Roles	**Dynamic**
No	[45, 46, 48, 49, 50, 51]	[17, 53]
Yes	[55, 56, 47, 41]	**this thesis**

Table 2.3.: Summary of the categorisations from Section 2.1 and Section 2.2.

approaches utilising adaptive scheduling or by mutual adaptation. Adaptive scheduling fully passes planning authority to the robot and diminishes the human agency in problem detection. The overarching objective of this thesis is to facilitate mutual adaptation, a concept that fosters dynamic teamwork and shared autonomy. However, even the two approaches from the literature falling into this category do not fully exploit concurrent dynamic teaming. The robot is either mostly idle or its sensor setup prevents it from perceiving human actions in time. Dynamic teaming with a high degree of concurrency has thus not yet been investigated or evaluated for collaborative assembly.

When teaming is enriched by computational SMMs, most approaches favour interaction scenarios with turn-taking. This results in low neglect tolerance and no concurrent activity. A further constraint on the approaches is that the majority of them use small task models with fewer than ten actions. There is a clear research gap: models that can be updated in dynamic non-interaction teaming scenarios for larger assembly tasks (Table 2.3).

To assess the cognitive ergonomics of a dynamic teaming approach that utilises SMMs, it is essential to select a suitable set of metrics. There are well-established metrics for cognitive workload and user satisfaction. However, evaluation practices for interaction quality, particularly fluency, are still emerging. Self-reports and productivity-related proxy metrics have been used in the literature to evaluate fluency for human-robot teaming. Two studies implementing human-robot interaction have found no clear link between fluency and completion time or efficiency. It should be noted that the two studies do not evaluate other metrics for cognitive ergonomics. It thus remains open whether the finding applies to teaming scenarios with high concurrency and whether overarching concepts could suffice to capture differences in fluency.

CHAPTER 3

Coordination in Human-Human Teams

This first technical chapter draws insights from human teams in terms of joint attention and shared mental models. It focuses on restrictions imposed by the limited sensing and understanding capabilities of state-of-the-art robot systems. The chapter presents the work from two laboratory studies, which have been published in the author's previous work [69]. The first study investigates joint attention in terms of hand gestures, facial expressions, and manipulative gestures (Section 3.1). The second study investigates the formation of shared mental models in terms of prediction capabilities (Section 3.2). Both studies rely on human subjects but restrict their interaction and perception capabilities towards what a robot would be capable of processing.

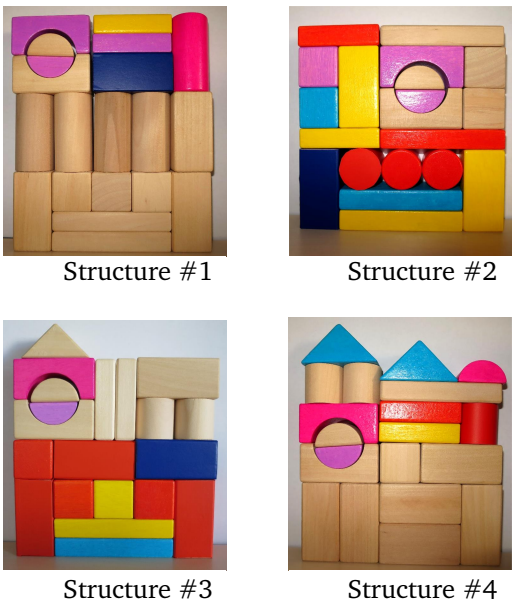
3.1. User Study on Non-verbal Communication

To investigate the effect of a lack of verbal communication on communication strategies, the study by Shah and Breazeal [70] was recreated. We then compare the results of our

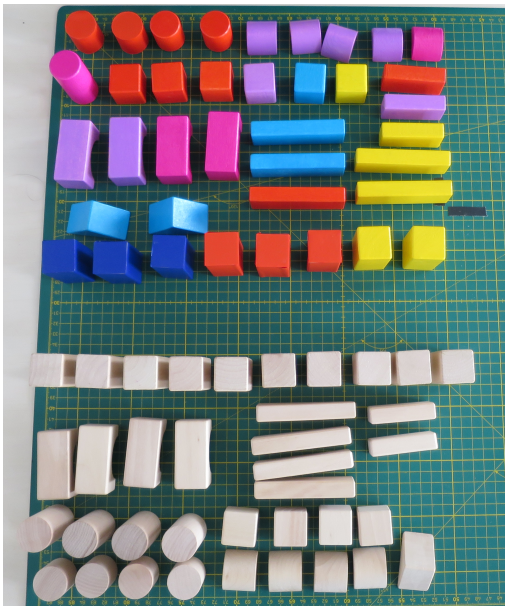
study with Shah and Breazeal [70] and Gleeson et al. [71]. Both cited papers conducted laboratory studies where two humans have to complete an assembly task with fixed roles. The study by Gleeson et al. [71] investigates non-verbal communication of missing knowledge. The task procedure used two-dimensional wooden shapes that had to be placed on a board to simulate assembly. During assembly, participants could not see each other's facial expressions, nor were they allowed to talk. Gestures were the sole means of communication. The primary conclusion of their research is that humans have a common dictionary of gestures to communicate certain instructions. We address the details when we compare the findings with our study in Section 3.1.2. Each participant only knew parts of the instructions to foster communication. In our vision, both partners should have full knowledge of the task. The question that arises from this is which gestures are preferred for coordinating tasks rather than communicating goals. This question is partly addressed in Shah and Breazeal [70]. They investigated team effectiveness as well as verbal and gestural communication with and without time pressure. Coordination was thus essential to complete it quickly. For the study, the setup was reconstructed as closely as possible to the original, but communication between the participants was prohibited. The precise setup and procedure are described in Section 3.1.1.

3.1.1. Procedure and Methodology

We briefly sketch the study procedure here. Details can be found in the author's prior work [69] and the appendix of this work in Appendix A. Figure 3.2 shows the setup of the study. On arrival, participants were randomly assigned to teams of two. Participants were then instructed and given some time to familiarise themselves with the task. This means they could practice any or all of the structures. Afterwards, the participants repeated the task three times (denoted as *trials*). The instructor ensured that participants stuck to the instructions and did not talk. In each trial, they had to build all the structures depicted in Figure 3.1a. Each participant was allowed to manipulate either tan or coloured blocks. However, there were not enough parts to complete all structures simultaneously (Figure 3.1b). Parts of previous structures had to be reused. Participants were informed that they could reuse parts, but not that the overall number was insufficient. The addition of manipulation and part constraints fosters communication during the first trial. At the end of the study, participants filled out a form with demographic information and comfort ratings. They were then compensated with 5 €.



(a) Participants build these structures during the study (based on [70]).



(b) Resources available to construct the structures. Note that there are only one short red block and four wooden cubes available.

Figure 3.1.: Resources and tasks of the study for non-verbal communication.

Figure 3.2.: Setup for the study for non-verbal communication. The execution is recorded in three ways (i) a top-mounted RGB-D camera records workbench state and hands motion, (ii) a camcorder records the participants (from the same angle as this image is taken), and (iii) the markers attached to the participants' hands are tracked by the red infrared cameras (OptiTrack Flex 3) to get precise and gap-free 3D motion

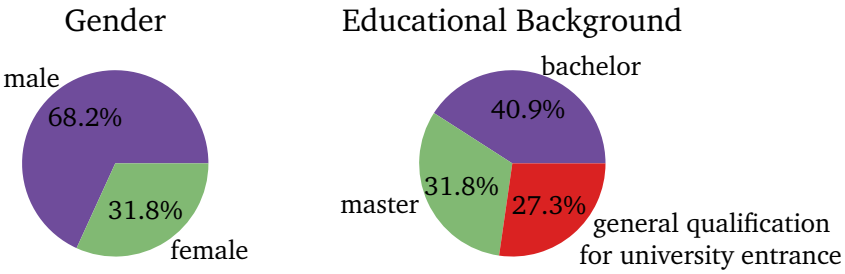
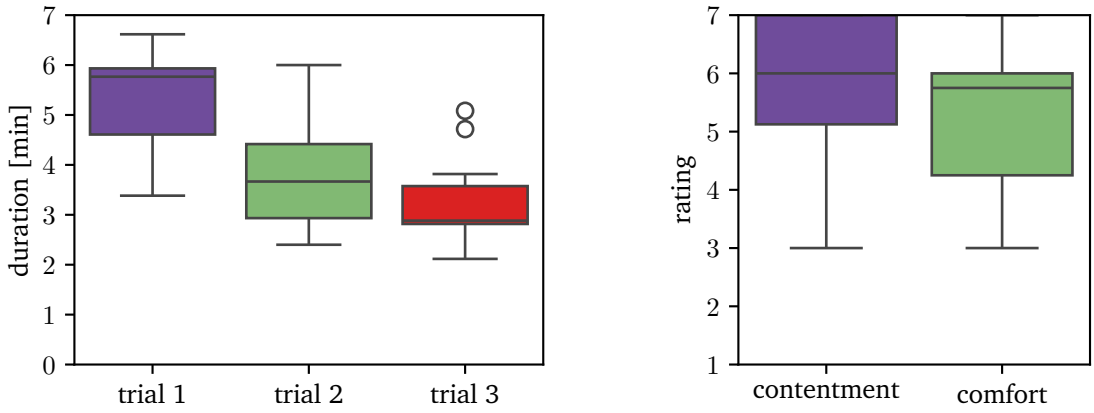


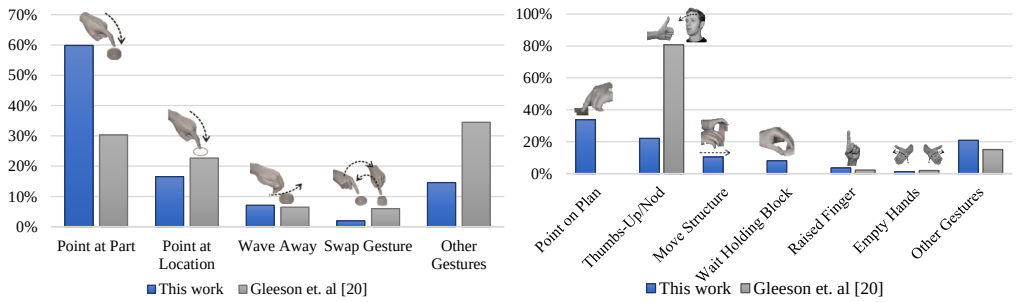
Figure 3.3.: Demographic information of the participants.



(a) Duration for each task completion.

(b) Affect ratings on a 7-point Likert scale.

Figure 3.4.: Objective and subjective measures of the first study.



(a) Object Manipulation **(b)** Team Coordination

Figure 3.5.: Comparison of relative usage of gestures for the two categories of object manipulation and team coordination.

3.1.2. Results and Discussion

A total of 22 participants (seven of whom were female) took part in the study. They were recruited from the local campus and thus had an academic background, as depicted in Figure 3.3. The average age was $24.5\,\mathrm{years}$. In our study, the average time required for participants to complete each trial was $247\,\mathrm{s}$. This finding aligns with the results reported in the study by Shah and Breazeal [70] with a duration of $240\,\mathrm{s}$. We thus conclude they found an equally efficient way of coordination (Figure 3.4a). A subjective evaluation of user satisfaction confirmed that comfort and contentment with performance are still high (average rating of 6 on a 7-point Likert scale: Figure 3.4b). The term comfort refers to the participants' 'level of comfort considering the disallowance to talk'. The term contentment refers to the 'level of contentment when performing the task (duration, coordination)'. Only two participants indicated that the lack of communication was a source of frustration.

During the study, participants were video recorded. Afterwards, the experimenter analysed the videos to identify communication events. Each communication was labelled with a small description. During the coding, an ad-hoc dictionary was generated. Categories were uninfluenced by taxonomies of other studies. Finally, top-level categories were formed, which we elaborate on in the following.

The primary means of communication are pointing gestures, followed by other hand gestures, manipulative gestures, and head gestures. Facial expressions and eye gaze only played a negligible role. A detailed breakdown of the frequency of gestures is given in Figure 3.5. In terms of object manipulation, the relative ordering is identical to the one found by Gleeson et al. [31]. Those gestures thus seem to form a common ground to non-verbally communicate object manipulation instructions. The conclusion is different

for team coordination. There, only the confirmation gestures seem to be universal. Besides these, our participants used new, context-specific gestures.The most frequent one is pointing to one of the structure images or even specific blocks to coordinate assembly and the next steps. Another notable gesture is the raised finger, which was rarely used in both studies, but conveyed important information of overwhelm or requiring a pause. Participants needed, on average, $38\,\%$ fewer communicative gestures for the second trial, leading to a reduction of $27.1\,\%$ in task duration as depicted in Figure 3.4a. The task duration is reduced further in the third trial. This decrease indicates that coordination and habituation are still ongoing after two trials.

However, not all task executions were perfect. From the videos, I identified three types of incidents where the participants' behaviour deviated from the intended one (Table 3.1). Given the frequency of invalid block placements, we conclude that it is

Frequency	Type	Description
$70\,\%$	Invalid Block	A block was placed at a location on the structure where it did not belong.
$33\,\%$	Early Decomposition	Participants started to dismantle the structure before completion. This could happen if they needed parts to complete another one, temporarily lifted a block to better fit in another one, or forgot that this one still needed to be completed.
$21\,\%$	Invalid Decomposition	Multiple blocks were taken at once, or parts of the structure collapsed.

Table 3.1.: Types of incidents where participants deviated from the intended procedure. Frequency is the fraction of trials where this type of deviation occurred.

advisable to give participants more guidance than a printed image. The issues with decomposition highlight the need for a task model that can cope with abrupt changes and reversal of actions. So far, we have investigated explicit and implicit communication. We further investigate prediction capabilities in these situations under the limited perception capacities of a robot system in Section 3.2.

3.2. User Study on Predictability of Human Assembly Patterns

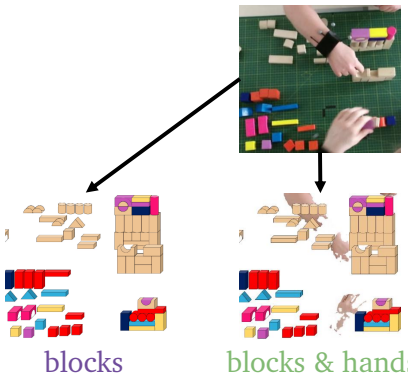
The perception capabilities of a robot are limited compared to the rich set of features a human can observe. As outlined in the survey by Fan, Zheng and Li [72], precise 3D pose information and occlusion remain major challenges. Current approaches for

object detection mainly return coarse bounding boxes and require an unobstructed view. Identifying small positional offsets or the object manipulated by the human is currently not feasible. Regarding the perception of human body position, current approaches merely provide approximate data. Therefore, subtle finger motions are not perceptible. These can e.g. indicate what the user is planning next, that the user is confused about something, or which object the hand is holding.

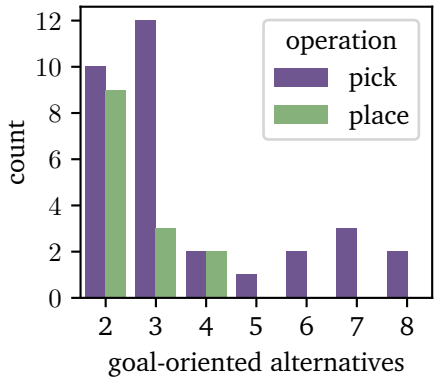
To simulate the perception capabilities of the robot, it is therefore necessary to limit the perceived information to what the robot gets as input. Since our study uses humans for the prediction, the question arises of how consistent their answers are. Prior research investigated the consistency of humans in terms of predicting verb-to-verb transition probabilities [73]. The verbs the authors included relate to everyday activities such as sleeping, walking, and eating, but also to instruction manuals. Key findings are that humans can accurately state action transition probabilities without knowing intentions or goals in advance. The study also shows that the inter-participant variation in terms of prediction accuracy is low (the $95\,\%$ confidence interval is at most $6\,\mathrm{pp}$ wide). Humans are, moreover, proficient in learning artificial transition probabilities [36]. This provides a promising ground for using humans as a replacement for a human-like robot and for measuring their prediction accuracy in terms of object-centric, discrete assembly actions limited by the perception capabilities of a current robot system.

3.2.1. Procedure and Methodology

To remove the information not perceivable by a robot, the videos recorded by the Kinect 2 are transformed into an abstract representation, as depicted in Figure 3.6a. The two layers of information (hands and blocks) are processed as follows. Hand positions are obtained by removing all non-skin regions of each image. This is achieved using standard video editing software through keying. Parameters were tuned to be more aggressive so that nothing besides the hands remains. This has the side effect that fingers are, in most cases, not fully visible—as intended. Representations of blocks are carefully designed to be well distinguishable. Then, those representations were placed according to their positions in the video while ensuring that the structures were fully visible. Pick and place events are signalled by making objects appear and disappear. The (dis-)appearance only occurred once the object's position was fully visible. Objects in transfer are not visible. The type of operation (pick or place), location and timestamp of an action in the video are denoted as an *event*. Two versions of the video are created: One that only includes the animated blocks and the other where blocks and hands are superimposed. The two



(a) Transformation of the video into an abstracted representation with reduced information.



(b) Characteristics of the videos: Total events grouped by goal-oriented alternatives. Events with only one alternative are not shown—thus, the depicted number of pick and place alternatives is not equal.

Figure 3.6.: Material used in the prediction study.

versions are referred to as *information levels*. In total, three scenes, each lasting 90 s to 120 s, were prepared. Each scene depicts a fraction of the construction process of one or two structures.

Five colleagues[1] participated in the experiment. The experiment was conducted at a workplace with the experimenter. The workplace had two screens to display the video, reference images of the completed structures and a text document to write down the predictions. At first, the participants were introduced to the structures that were supposed to be built in the videos. The reference images contained unique labels for each block position to ease identification. The participants were given time to understand the initial arrangement of blocks and the completion state of the structure, and they could think about the next steps that needed to be taken. The video was played at half speed so that participants had time to perceive and reason about the provided information. They could pause and resume the video at any time to write down their prediction of what the next pick or place action is. The experimenter ensured that (i) each record included the timestamp of the video, (ii) no record was modified later, and (iii) the effect of the prediction, e.g. the placement of a block, had not yet been displayed in the video. The free form of a text file was chosen so that participants could express all kinds of additional thoughts they had, e.g. certainty of prediction, alternatives, rankings. Participants were

[1] It was opted for a small convenience sampling because inter-participant variance seems to play a minor role [73] and the participant selection ensured that all participants understood and conducted the study thoroughly.

not forced to make predictions at certain times. Each participant watched and annotated all three scenarios, but their order and information level were chosen at random.

3.2.2. Results and Discussion

Before analysing the results, one needs to preprocess the data into a standardised form, define what correct predictions are, and relate them to the number of alternative actions. We use the number of alternative actions to later group actions. First, we need to specify when we consider two actions as identical or different. For instance, participants were allowed to place resources anywhere in the workspace. We therefore treat the location on the table as irrelevant. On the structures, however, positions matter. For instance, in Structure #1, it matters whether one places the wooden upright block on the left or the right because it enables different follow-up actions. We therefore distinguish pick actions from the workplace by block type and those from structures by each block. From those actions, some are not reasonable to do, e.g. picking a block that cannot be placed on any structure. We thus only consider the action set that contributes to the overall goal, referred to as *goal-oriented*: (i) Blocks can only be placed on unfinished structures. (ii) Picking is only allowed for block types which have a goal-oriented place action. (iii) Picking from structures is not allowed—unless resources must be reused to proceed. These constraints reduce the number of alternatives by a factor of two. The number of goal-oriented alternatives was manually annotated for each event. A formal algorithm to compute goal-oriented actions is later given in Section 5.5. Figure 3.6b shows the difficulty of the prediction tasks. In most cases, only two to three alternatives existed. For pick actions, up to eight alternatives exist if there are eight different block types with which to continue the structures.

The analysis proceeded as follows: First, the experimenter coded the predictions in a standardised form, including timestamp, operation, and location. All participants' predictions are then labelled as either correct or false. For each ground truth event, we determine whether each participant stated a prediction before the event (*predicted*) or not (*unpredicted*). We do this as follows: A prediction is labelled as *correct* for a ground-truth event if (i) the operation and location match, and (ii) no more than $6\,\mathrm{s}$ pass between the prediction and the event. However, if participants predicted a place action before a block of that type was picked, it counted as a correct prediction of a pick action, too. All remaining predictions were labelled as *false*. False predictions are also matched to ground truth events using the same time threshold as correct predictions. All ground truth events matched to a correct or false prediction are counted as predicted. All others

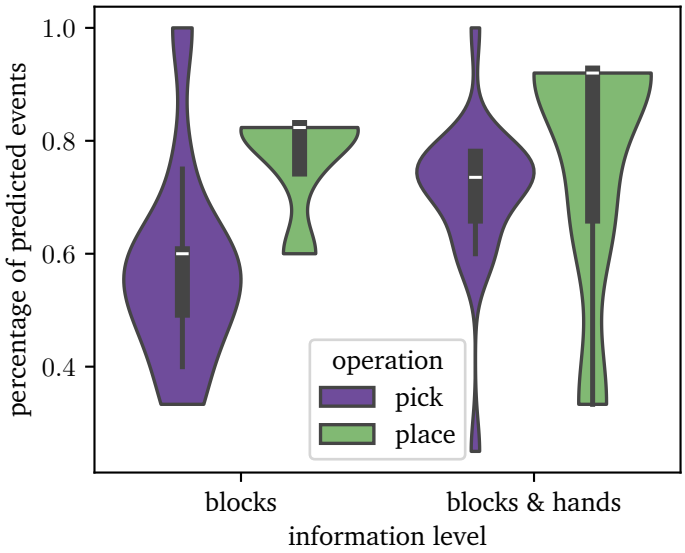


Figure 3.7.: Percentage of events for which participants stated a prediction. Higher values indicate better performance.

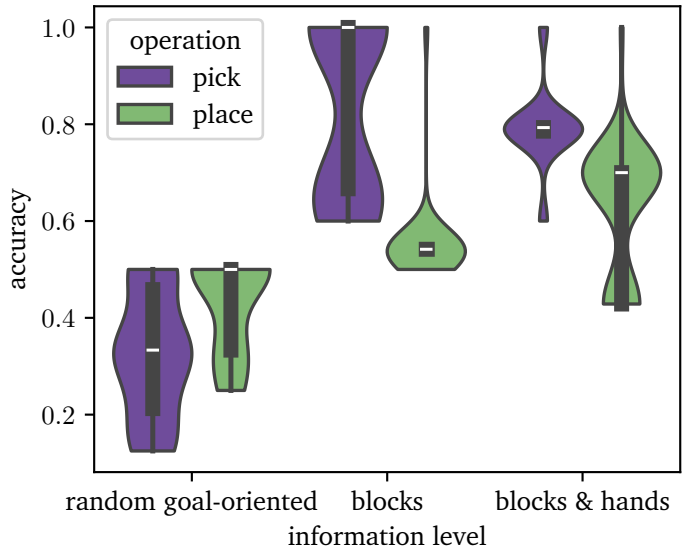


Figure 3.8.: Accuracy of predicting the next action when only blocks are visible (blocks) or the position of hands is added (blocks & hands). Higher values indicate better performance.

are counted as unpredicted. Since participants were not forced to make predictions, the first question is: Are there sufficient predictions given the number of ground truth events? Figure 3.7 answers the question. It displays the ratio of predicted to all events. The distribution is obtained by grouping and weighting the predicted events by the number of goal-oriented alternatives before calculating the ratio. Situations where only one goal-oriented action is possible were excluded from the analysis. The plot shows that in most cases, predictions cover at least $60\,\%$ of the events. Pick actions with many alternatives have a very low percentage of predicted events. These occur at the beginning of the video sequence, when participants are still developing a shared mental model. In such cases, participants often expressed uncertainty regarding the subsequent steps and preferred to refrain from making predictions.

The accuracy of predictions, measured as the ratio of correct predictions divided by all predictions, is depicted in Figure 3.8. The bimodal distribution of pick predictions in the blocks information level is due to the low number of predictions, where a few small groups with $100\,\%$ accuracy could create this peak. Besides the information levels, Figure 3.8 presents the accuracy one achieves when selecting a random action from the goal-oriented alternatives, denoted as *random goal-oriented*. To generate the plots for random guessing, accuracy is calculated as one divided by the number of alternatives and weighted by the number of events with that many alternatives. For both information levels, prediction accuracy is far better than random guessing ($p < 1 \times 10^{-8}$ using a pairwise Wilcoxon rank-sum test). The results of all tests are presented in Appendix A.2. The difference for place predictions comparing the information levels blocks and blocks & hands is also significant ($p < 0.017$). Only the pick predictions show no significant difference when adding hand position information. The addition of hand positions thus only provides some benefit if we have multiple well-distinguishable target locations. For both information levels, the median accuracy of pick predictions is higher than for place. This indicates that participants had a good intuition—when they made a prediction—which type of block would be used next.

Two uncontrollable factors influence the accuracy. The first one is that the accuracy calculation only includes predicted events. We do not know how well participants would have performed if they had been forced to make predictions before each ground truth event. The second factor is the attention of the participants. They needed to pay attention to where blocks appeared and disappeared (but could rewind to check). Moreover, they needed to figure out by themself which actions are available. The latter effect could also be positive by not distracting people when explicitly showing less relevant, goal-directed alternatives.

3.3. Conclusions and Discussion

This chapter described two laboratory studies to investigate gesture-based team coordination and the formation of shared mental models. The first one investigated human-human teaming in a cooperative assembly scenario. The second study focused on non-verbal communication strategies and performance metrics.

In terms of communication strategies, the first study revealed a set of frequently used gestures to communicate object manipulation. The two most prominent gestures are pointing at an object and pointing at a location. This finding is in line with related research and shows the broader applicability of the gesture dictionary. While implementing the other parts of the software framework, it turned out that the incorporation of gestures requires far more research effort. The main hindrances are the robust detection of fingers, as pointed out in Section 4.6. I therefore decided to leave gesture recognition and processing open as an expansion option for the future.

In terms of performance metrics, participants achieved the same completion time as in a comparable study where they were allowed to talk. This motivates us to test a human-robot teaming setup without verbal or textual feedback in either direction. Another finding from the study is three types of errors and deviations in human-human teaming. These are the invalid placement of a block, followed by invalid and early decomposition. In the following chapters, we introduce two measures to account for these errors: (i) a robust state tracking that includes reversal of actions and (ii) more visual guidance on object placement. The first aspect prevents the system from being trapped in incorrect task states. This shall ensure that trials of the human-robot study described later in Section 7.3 require fewer interventions by the instructor. The second aspect brings study participants closer to the envisioned worker who precisely knows how to execute the task. Placement mistakes should therefore occur less frequently. The visual guidance is, however, not designed to fulfil the purpose of a full-fledged worker instruction or assistance system.

The second laboratory study investigated how predictable task execution is. Recordings from the first study were taken and preprocessed such that only object locations and hand positions were visible. Still, humans can predict the next actions of the task execution. They achieve a prediction accuracy of over $60\,\%$, no matter how many alternative actions exist. This is a big difference compared to picking an action at random. Thus, even with the limited information of block positions, people can recognise the shared mental model underlying the task execution and can predict the next actions. Adding the information of hand positions further improves prediction accuracy and increases the chances that

people feel confident enough to make a prediction. We conclude that block positions and occurrences yield a good starting point to investigate shared mental model formation in human-robot cooperation. The incorporation of additional information, such as hand trajectories, has the potential to enhance the accuracy of predictions. We therefore investigate hand tracking as a means of narrowing down the search space of actions in Chapter 5.

CHAPTER 4

Robust Hand Tracking

Tracking the human position can give valuable information about which actions could have been executed—even if the granularity or reliability of information is insufficient to identify individual actions, see e.g. [17]. While Riedelbauch [17] uses LiDAR to track the human's legs, this is insufficient in our case, where the human mainly remains stationary. To gain insights about executed actions, the hand motion is relevant. We start by reviewing related literature and introducing notations.

4.1. Related Work

Many approaches estimate the movement of the hand wrist in conjunction with the whole body [74]. 'Three-dimensional *human pose estimation* involves estimating the articulated 3D joint locations of a human body from an image or video' [74]. Human pose estimation can be achieved by (i) inertial measurement units (IMU), (ii) by colour

segmentation, (iii) by skeleton tracking in camera images, (iv) with data gloves, (v) or with dedicated markers attached to the body and captured by a surrounding detection system [19]. Without additional global position information, IMU suffer from drift over time. Data gloves and markers are among the most precise systems. However, both severely restrict sensitivity and manoeuvrability. Tracking approaches based on colour images are less invasive and distracting. On the other hand, they suffer from lower precision, occlusion[1] and jerkiness.

As this work focuses on fluent human-robot teaming at assembly stations, non-intrusiveness and occlusion are important to consider. I therefore opted for tracking mechanisms based on colour images. Established frameworks for human pose estimation are OpenPose [75], MediaPipe [76], and the Kinect SDK [77]. The skeleton tracking of the Kinect cameras suffers a lot when the lower part of the human body is occluded [77]. Unfortunately, this is the default case when a human sits at an assembly station. We therefore focus on frameworks that can cope with tracking the hands or arms only.

'*Hand pose estimation* corresponds to estimating all (or a subset of) the kinematic parameters of the skeleton of the hand' [78]. *Hand tracking* is the process of calculating consecutive hand poses over a period of time. The output data of hand tracking are *hand trajectories*, i.e. a sequence of hand poses at consecutive points in time. Besides the above-mentioned frameworks for human pose estimation, dedicated ones for hand tracking exist. These are cameras with integrated hand tracking, such as the Leap Motion [79], or head-mounted displays, such as the HoloLens 2 [80] or Meta Quest 3 [81]. The Leap Motion focuses on short-distance mid-air hand tracking. It thus struggles when the hand is close to an object or when grabbing the object. Detection accuracy at medium distances is considered bad [82]. The HoloLens 2 offers good tracking accuracy with a positional offset of around $2.5\,\mathrm{cm}$ at the edge of the vision field [83]. Accuracy improves when the hand is bigger or in the centre of the field of vision. Moreover, the HoloLens 2 robustly tracks the hand while the user manipulates an object. The Meta Quest 3 was released after all experiments for this thesis had been conducted.

The capabilities for hand tracking and plan communication (as later motivated in Section 7.3) motivated the inclusion of the HoloLens 2 for the overall setup. Still, the hand trajectory has gaps when the user reaches out of the vision range. This happens, e.g. when the user checks the next step while picking up a far-away object. Another issue is that the hand model exhibits some jerkiness, where single fingers or the whole hand

[1] Skeleton tracking by camera images usually employs a single camera. Marker based setup often use multiple cameras and are thus less prone to occlusion. However, if a marker gets out-of sight, its tracking might be lost until a re-calibration is done.

abruptly change position within consecutive frames. To cope with these issues, this work explores further approaches for hand tracking.

Methods for hand pose estimation can be categorised into model-driven, data-driven, and hybrid. 'Model-driven methods generate hypothetical hand poses and compare them with the observations retrieved from depth cameras' [84]. In the end, an optimisation problem must be solved where the problem space is all hand poses and the objective function is derived from the discrepancy between the generated hand pose and the observed depth map.

Model-driven approaches require a good initial estimate of the hand pose—usually taken from the previous frame. They often suffer loss of tracking in the case of fast hand movement [84]. 'Data-driven methods learn a direct mapping from the observations to a discrete set of the annotated hand poses' [84]. The output of data-driven methods is often a heatmap for each joint. The maximum of the heatmap is then the position of the joint in the 2D space of the input image, referred to as a *keypoint*. They are robust to previous inaccuracies and can recover from the loss of tracking. Hybrid methods use data-driven ones as an initialisation and then refine the pose using a model-driven method. Li, Liu and Tan [84] identify three key aspects where each of the method classes has its individual strengths and weaknesses:

1. **Temporal coherence** refers to the continuity of the hand pose over time. It prevents sudden changes between neighbouring frames. Many of the model-driven methods use temporal priors to initialise the pose estimation. However, only some methods of each category explicitly ensure temporal coherence for the calculated pose. Li, Liu and Tan [84] list 14 references that do so (e.g. [85, 86, 87]). Unfortunately, only the source code for the approach by Tkach, Pauly and Tagliasacchi [88]. However, their method requires a blue wristband and thus only works for a single hand per image.

2. While data-driven approaches show good **runtime performance**, hybrid and model-driven approaches suffer from higher calculation demands to solve the optimisation problem.

3. Enforcing **anatomic and kinematic constraints** prevents the generation of impossible hand poses. Data-driven approaches cannot guarantee this. Model-driven approaches utilise modelling objects ranging from geometric primitives (e.g. [89]) over sphere-meshes (e.g. [88]) to mixtures of Gaussian functions (e.g. [90]) to model the hand surface. All of them can detect when parts of the hand would prune each other and discard the pose.

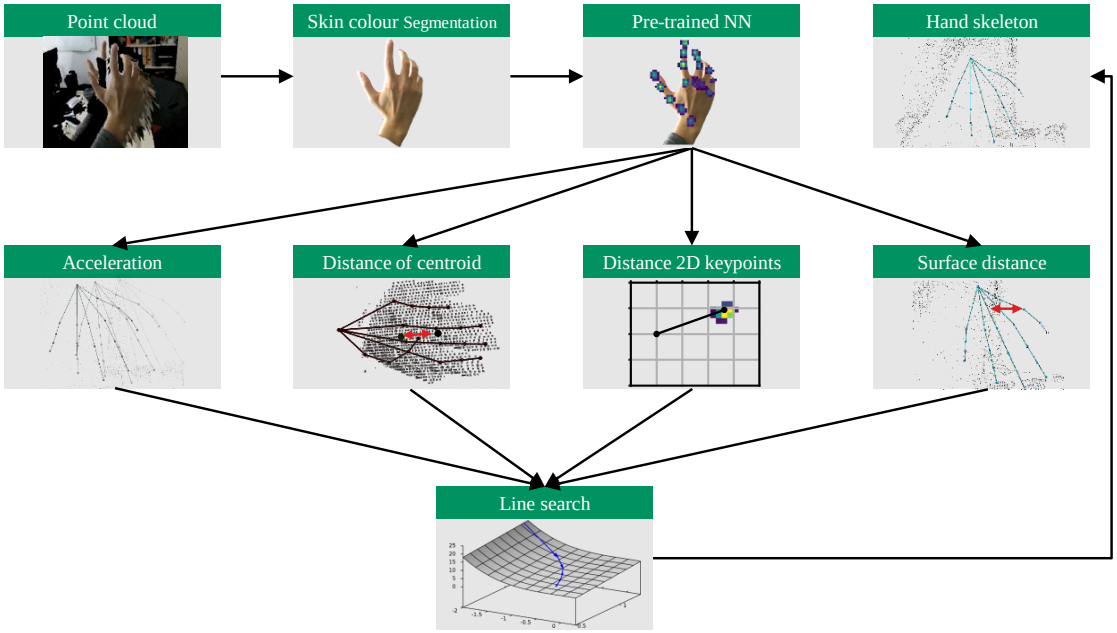


Figure 4.1.: Hand tracking pipeline. The top row shows the processing steps. The second row shows the components of the error function described in Section 4.5

The literature review reveals numerous approaches to hand-tracking, yet only a subset provide source code or software frameworks. Among these, none fully meet our requirements—specifically, tracking multiple hands at a distance of $1\,\mathrm{m}$ to $1.5\,\mathrm{m}$. Therefore, the implementation of a new hybrid framework for hand tracking is required. The implementation takes inspiration from methods in the literature and reuses components wherever possible. It is designed for a high degree of parallelisation but can also cope well with limited computing resources, e.g. when other parts of the software require more computational resources. A novelty is the introduced hand model that ensures anatomic constraints by kinematic constraints while only putting small restrictions on the space of anatomically valid poses. We start by introducing the processing pipeline before addressing segmentation, data-driven pose estimation, and model-driven pose refinement. The implementation assumes numerical values in standard SI units (meter, rad, etc.). At some points we use a better readable notation, e.g. $1\,\mathrm{cm}$ or $45°$. This translates to numerical values of 0.01 (m) and 0.7853981634 (radians).

4.2. Processing Pipeline

The overall setup assumes a camera with an integrated RGB and depth sensor mounted above the workspace. We assume that the extrinsic and intrinsic camera parameters (see [91] for terminology) are known after performing camera calibration. Appendix B.1 provides further details. That means, we have—ignoring lens distortion—two projection matrices P_c, P_d from the workspace coordinate system into the image space. Here, P_c refers to the projection matrix for the colour measurement unit of the camera and P_d for the depth measurement unit. Due to the spatial offset of those two units, a single projection matrix is insufficient. We elaborate on the problems associated with this spatial offset later. Each frame received from the camera consists of two data structures: an RGB image and a coloured point cloud processed with the Point Cloud Library (PCL) [92]. The point cloud is preprocessed to remove everything below the workspace surface. Figure 4.1 provides an overview of the subsequent processing pipeline. The key aspects are:

Skin Segmentation and Tracking identifies image parts with skin-like regions and matches them to hand trajectories. (Section 4.3)

Hand Model and Pose Estimation obtains a pose estimation for a new skin segment. (Section 4.4)

Pose Refinement employs a bounded line search to find the best pose with regards to a multi-objective optimisation function (Section 4.5)

Skin segmentation deploys a standard algorithm [93] to find connected subsets of the image with skin colour, denoted as *skin segments*. Skin segments are then matched to hand trajectories based on proximity to the last pose. Once the segment is associated with a hand trajectory, an **estimate of the hand pose** needs to be obtained. The camera image is cropped to the skin segments and then processed by a data-driven method. For the latter, freely available pre-trained neural networks that output keypoints are used. The keypoints combined with the depth information are used to calculate the pose of the hand palm. All further steps make use of the hand model introduced in Section 4.4. Finally, **finger poses and hand positioning are refined** based on the output of the hand pose estimation and the depth image. This is achieved by formulating and minimising several error functions.

The pipeline requires processing large amounts of data with low latency. The pipeline is therefore split into small chunks of work. This allows a high degree of parallelisa-

Description	Parameter
Threshold for definite skin segment b_d	$70\,\%$
Threshold for likely skin segment b_l	$15\,\%$
Minimum hand length	$20\,\mathrm{pixel}$
Minimum hand area	$0.3\,\%$ of the image
Thickness of a hand t_h	$2\,\mathrm{cm}$
Outlier distance threshold d_o	$15\,\mathrm{cm}$ from centroid[3]
Kernel for morphological operations	Circle with a diameter of $5\,\mathrm{pixels}$
Search radius for normal estimation used in [92]	$1\,\mathrm{cm}$
Lower clamp value for IoU t_{IoU}	0.24
Threshold t_s for the similarity score of two bounding boxes	$70\,\%$

Table 4.1.: Parameters for skin segmentation and tracking. Most of the parameters are heuristically chosen when coding and testing the hand tracking.

tion, though only four threads are used in total to have enough CPU resources for the other software components. Two threads are allocated to skin segmentation[2], one for evaluating the neural network and the remaining two to run the line search for each hand.

4.3. Skin Segmentation and Tracking

The first step when a new camera frame arrives is to identify skin segments. We use the Bayesian classifier described in Argyros and Lourakis [93]. The input image is first converted to the YUV colour model, and the Y-component is discarded. This makes it more robust to illumination changes. Given a UV colour value $c \in [0, \ldots, 255]^2$, the classifier requires a probability estimate $P(s \mid c)$—the conditional probability that the pixel is a skin pixel s given its colour value c. The probability $P(s \mid c)$ is calculated from training data of skin and non-skin images. Manually recorded and cropped images of different persons and backgrounds are used to calculate the probability for all colour values. All images for training were taken with the Kinect 2 camera using the same setup as for the other studies. The classifier proceeds as follows:

1. Calculate skin probability for each pixel [93]

[2] where one is a helper thread to distribute the work of calculating properties for each skin segment

[3] roughly $\frac{2}{3}$ the length of a hand

2. Generate two binary images b_l, b_d from the skin probabilities using the thresholds in Table 4.1

3. Apply dilation to b_d[4]

4. The intersection of b_l and the hand areas from the previous frame are added to b_d

5. Connected components are calculated, and those discarded that do not match the criteria for a hand, as defined in Table 4.1

6. The remaining connected components are filled, resulting in a skin segment $S \subset \mathbb{N}^2$ and returned.

The skin segmentation outputs skin segments, which are pixel indices that likely belong to hands. To obtain the corresponding points of the point cloud, the spatial offset between the colour and depth measurement units of the camera must be taken into account. The offset causes *parallax errors* (Figure 4.2a), where different depth points can project onto the same RGB pixel. Consequently, colour information is not available for some depth points. Another issue is laser shadows produced by the depth measurement unit. *Laser shadows* occur when the camera smooths the point clouds and creates non-existent points between sharp-edged boundaries (see Figure 4.2b and [95] for a detailed explanation). In the following, we explain how to handle parallax errors and laser shadows.

A point $p \in \mathbb{R}^3$ from the point cloud belongs to a skin segment $S \subset \mathbb{N}^2$, if P_c projects p into a skin segment, i.e. $P_c \cdot p \in S$. Assume P_c projects depth points $p_0, \dots p_n \in \mathbb{R}^3$ onto the same pixel inside a skin segment. Then, the following depth points are disregarded as background: $\left\{ p_i \;\middle|\; \|p_i\| > \min_{j=0,\dots n} \|p_j\| + t_h \right\}$ (Table 4.1). With that condition, we obtain all depth points belonging to a skin segment. We further filter outliers more than d_o (Table 4.1) away from the centroid. Those filtering steps avoid adding depth points from the background or laser shadows. References to the depth points are then stored in a quadtree[5] using the pixel coordinates of that point in the colour image as the key. This eases the search for the closest 3D point given some pixel coordinates. Moreover, we have the guarantee that the returned point is always part of a hand. We later refer to the

[4] b_d underestimates the hand area to reduce false positives but this can lead to parts of the hands belonging to separate segments. To combine those segments, dilation is applied. Wherever possible the OpenCV [94] library is used.

[5] `https://web.archive.org/web/20200901000000*/https://pointclouds.org/documentation/group__kdtree.html`

(a) The parallax error of depth and colour camera leads to the colour information of index finger and thumb being projected onto the wall as well. Black regions indicate that there is no point cloud data.

(b) Laser shadows around the index finger. The bumps towards the left are created by points located between the wall and the finger. The ones to the right stem from artefacts where points are closer to the camera than any real object.

Figure 4.2.: Errors of coloured point clouds. Point clouds are represented by a coloured triangular mesh instead of points.

data structure as the *surface point lookup*. For each surface point, we estimate a normal vector. The calculation uses the implementation in PCL[6] [92] (Table 4.1).

This concludes the construction of skin segments per frame. In the next step, skin segments must be matched to hand trajectories so that the new skin segments add a new time step to the trajectory. The matching is based on the proximity of their bounding boxes, taking motion into account. The expected position of a segment is the linear extrapolation of the previous positions. The closest segment-hand pairs are matched in a greedy way.

To do that, a similarity value is calculated for each pair of skin segment and hand. The similarity calculation takes position offset and overlap into account. The calculation requires the axis aligned bounding box (denoted by b_{-2}, b_{-1}, b_0 for the three most recent frames) and the centroid (denoted by c_{-2}, c_{-1}, c_0 for the three most recent frames). For the hand, both are calculated based on the kinematic model described in Section 4.4. For the skin segment, the point cloud is used. Given the centroids $c_a, c_b \in \mathbb{R}^3$ and two axis-aligned bounding boxes $b_a, b_b \subset \mathbb{R}^3$, their similarity is given by

$$s(b_a, b_b, c_a, c_b) = (c_a - c_b) \cdot \left(1 - (1 - t_{\text{IoU}}) \cdot \frac{\text{vol}\,(b_a \cap b_b)}{\text{vol}\,(b_b)}\right) \tag{4.1}$$

where $\text{vol}(\cdot)$ denotes the volume of the bounding box. The similarity is inspired[7] by the Intersection over Union (IoU) measure [96]. The IoU is then inverted and clamped to $[t_{\text{IoU}}, 1]$ (right factor in Equation 4.1) before multiplying it with the position offset (left factor in Equation 4.1). Since the bounding boxes are more affected by outliers, we give higher priority to the difference of the centroids. That's why the inverted IoU is clamped to roughly one-fourth and is not allowed to become zero.

Let $\text{bell}(x, \sigma) = \exp\left\{-x^2/(2 \cdot \sigma^2)\right\}$ denote the bell curve—to be precise, the non-normalised Gaussian distribution with a standard deviation of σ, the mean at zero, and a maximal value of 1—and $\text{diagonal}(\cdot)$ the length of the diagonal of a cuboid. Then, the overall similarity is calculated:

$$\max \left\{ \begin{array}{c} \text{bell}\left(s(b_0, b_{-1}, c_0, c_{-1}), \frac{1}{3} \cdot \text{diagonal}(b_{-1})\right), \\ \text{bell}\left(s\left(b_0, 2 \cdot b_{-1} - b_{-2}, c_0, 2 \cdot c_{-1} - c_{-2}\right), \text{diagonal}(b_{-1})\right) \end{array} \right\} \tag{4.2}$$

[6] https://web.archive.org/web/20200801000000*/https://pointclouds.org/documentation/classpcl_1_1_normal_estimation.html

[7] The denominator does not use the union over b_a and b_b since parameter b_a can become very large due to outliers (in later calls of this function, b_a is always the bounding box of the skin segment).

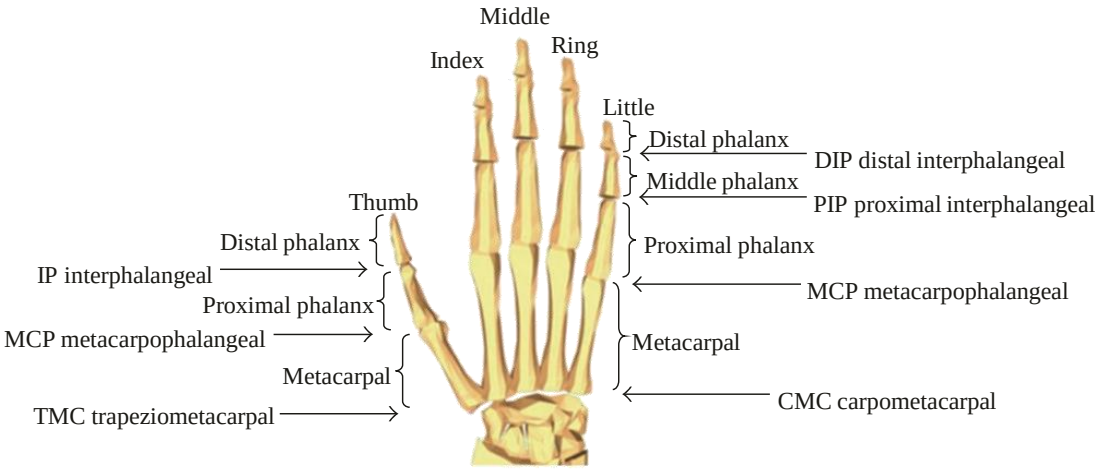


Figure 4.3.: Anatomical details of the hand [97]

The first component of the max operation compares the current and previous bounding box, giving higher similarity if they are closer and overlap more. The second component performs a linear extrapolation of the hand trajectory and compares the resulting bounding box with the current one. Both components are needed to find good matches for fast-moving and quickly stopping hands.

All skin-hand pairs are ordered by similarity and matched in descending order. The matching stops if all hands have gotten a match or the similarity drops below the threshold t_s (Table 4.1). Already matched hands are skipped. Multiple hands can be matched to the same skin segment S. In that case, we further subsegment S by moving the 2D bounding boxes into the bounding box $b_s \subset \mathbb{N}^2$ of S. We first linearly extrapolate the hand trajectory using the hand model described in Section 4.4. The resulting hand pose is then projected into the coordinate system of the colour image and encapsulated by an axis-aligned bounding box $b_h \subset \mathbb{N}^2$. The bounding box b_h is truncated to the size of b_s if it is larger, and the subsegment bounding box is given by $b_h + \arg\min \left\{ \|t\| \;\middle|\; t \in \mathbb{Z}^2 \text{ and } (b_h + t) \subset b_s \right\}$.

4.4. Hand Model and Pose Estimation

Hand poses can be modelled in terms of 26-degree-of-freedom (26-DoF) skeleton models (e.g. [84, 98, 89, 99]). Six degrees of freedom are attributed to the pose of the wrist in 3D space. Each finger has four independent degrees of freedom. Two are for the metacarpophalangeal joint, and two are for bending the finger (see Figure 4.3 for termin-

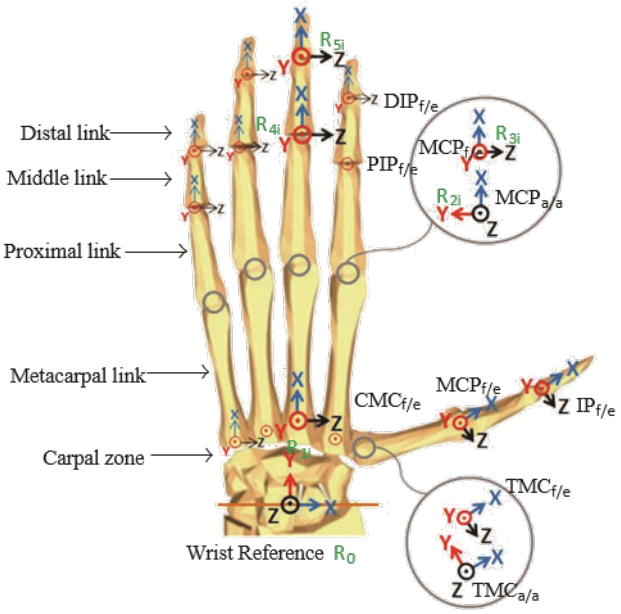


Figure 4.4.: Local reference coordinate system of the hand and descendant joints. [97]

ology). The full degree of freedom model allows overlapping fingers and does not take coupled motion into consideration. For example, the proximal and distal interphalangeal joints are bent together in a ratio of 3:2 to 2:1. The same holds for the adduction and abduction of all fingers except the thumb [97].

We use a more restricted model with 18 degrees of freedom, which fits most use cases but prevents unnatural or impossible poses: The interphalangeal joints are combined in one parameter. The spreading of the fingers is another parameter fused from the abduction of the metacarpal joints. The spreading parameter indicates the angle by which the index finger and the small finger deviate from being parallel. The thumb has three degrees of freedom: one for adducting the trapeziometacarpal, one for bending the trapeziometacarpal and one for the combined flexion of the carpometacarpal and interphalangeal joints. Limits for maximum and minimum joint angles are enforced. Together with a reduced degree of freedom model, this ensures that all encoded poses are plausible, such that fingers do not penetrate each other or the palm. Each joint—the origin of its local coordinate system, to be precise—plus the fingertips define the 21 keypoints of the hand. Keypoint indices start with 0 at the wrist, 1 to 4 for the thumb, and then index from the metacarpal joint to the tip from index to little finger.

Figure 4.4 depicts the coordinate systems for the hand model. Placement of the coordinate systems and terminology follows the modified Denavit-Hartenberg parameters

Finger	Metacarpal $a_0(i)$	Proximal p. $a_1(i)$	Middle p. $a_2(i)$	Distal p. $a_3(i)$
Thumb ($i = 0$)	$3.50\,\mathrm{cm}$	$4.00\,\mathrm{cm}$	$3.25\,\mathrm{cm}$	$0.002\,\mathrm{cm}$
Index ($i = 1$)	$8.84\,\mathrm{cm}$	$3.7\,\mathrm{cm}$	$2.48\,\mathrm{cm}$	$2.17\,\mathrm{cm}$
Middle ($i = 2$)	$8.45\,\mathrm{cm}$	$4.64\,\mathrm{cm}$	$3.14\,\mathrm{cm}$	$2.60\,\mathrm{cm}$
Ring ($i = 3$)	$8.32\,\mathrm{cm}$	$4.18\,\mathrm{cm}$	$2.59\,\mathrm{cm}$	$2.00\,\mathrm{cm}$
Little ($i = 4$)	$8.00\,\mathrm{cm}$	$2.67\,\mathrm{cm}$	$1.90\,\mathrm{cm}$	$1.97\,\mathrm{cm}$

Table 4.2.: Segment lengths of the fingers derived from the normalised hand model in Mueller et al. [99] and average hand length [97]. In the table head, phalangeal is abbreviated by p.

[100]. In the following, a_i denotes the link length, θ the joint angle. Link twist and offset are fixed values in the following equations. The wrist reference coordinate system is rooted in the wrist with z perpendicular to the palm, y pointing towards the fingers and x to the right. The transformation from the metacarpal coordinate system to the wrist reference coordinate system for finger i is as follows:

$$T_{(a_0(i)\cdot\sin\theta_0(i),a_0(i)\cdot\cos\theta_0(i),0)} \cdot R_z(90°) \cdot R_z\left(\delta_s(i) \cdot \theta_1(i)\right) \tag{4.3}$$

where $R_z(\theta)$ is an affine rotation matrix around the z-axis with angle θ and T_d is a translation matrix with vector d. For the thumb, an extra rotation is introduced:

$$T_{d_0} \cdot R_z(45°) \cdot R_x(90°) \cdot R_z\left(\theta_1(0)\right) \tag{4.4}$$

where

$$d_0 = R_z(-\theta_0(0)) \cdot R_x(-30°) \cdot (0, a_0(0), 0)^T \tag{4.5}$$

If finger spreading is 0°, all fingers are parallel, and the y-axes of the coordinate systems of the metacarpophalangeal joints align with the y-axis of the wrist coordinate system. To transform from the fingertip to the coordinate systems of the metacarpophalangeal joint:

$$R_x(90°) \cdot T_{(0,a_1(i),0)} \cdot R_z(0.6 \cdot \theta_3(i)) \cdot T_{(0,a_2(i),0)} \cdot R_z(0.4 \cdot \theta_3(i)) \cdot T_{(0,a_3(i),0)} \tag{4.6}$$

The combined angle of the last joints is split and attributed $60\,\%$ to the proximal interphalangeal joint and $40\,\%$ to the distal interphalangeal joint. To get a right hand, a reflection on the y-z plane is added as the left-most transformation step. The length of segments and joint limits are taken from Chen et al. [97] and listed in Table 4.2 and Table 4.3 for reference.

Finger	Thumb	Index	Middle	Ring	Little
$\theta_0(i)$: radial distance of metacarpals	$59.38°$	$19°$	$5.4°$	$-5.8°$	$-21.1°$
$\delta_s(i)$: propagation factor for θ_1	-	1	$\frac{1}{3}$	$-\frac{1}{3}$	-1
$\theta_1(i)$: thumb adduction / finger spreading	$[-50°, 20°]$		$[-4.5°, 10°]$		
$\theta_2(i)$: metacarpophalangeal joint flexion/extension	$[-40°, 20°]$	$[-90°, 20°]$	$[-90°, 20°]$	$[-90°, 20°]$	$[-90°, 20°]$
$\theta_3(i)$: proximal + distal interphalangeal joint flexion/extension	$[-160°, 25°]$	$[-200°, 0°]$	$[-200°, 0°]$	$[-200°, 0°]$	$[-200°, 0°]$

Table 4.3.: Angular limits of the joints.

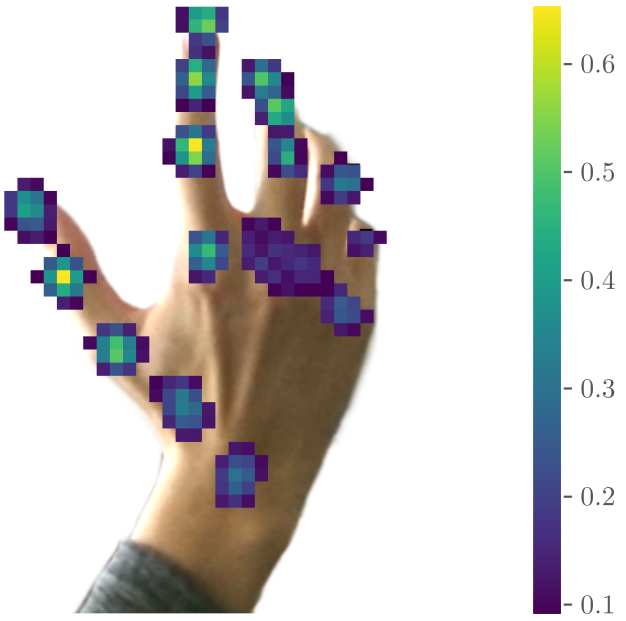


Figure 4.5.: Heatmaps of all 22 keypoints stacked onto the input image. Values below 0.09 are transparent. Due to the hand pose, the heatmaps for the tips of the little and ring fingers have no peaks. The output heatmaps have a lower resolution than the input image.

For the data-driven method, two ready-to-use models have been considered: The hand keypoint detection from OpenPose by Simon et al. [101] and GANerated hands by Mueller et al. [99]. Both output heatmaps for each joint from which the keypoint locations can be derived. None of them showed superior robustness in terms of keypoint detection. I thus opted for GANerated hands, which additionally outputs the 3D coordinates of each joint relative to the wrist (hereinafter referred to as *estimated joint coordinates*). Figure 4.5 presents an example output.

An initial estimate of the new hand pose is obtained by algebraic calculations. The calculation is subdivided into the pose of the palm and the joint angles of the fingers. Joint angles of the fingers are calculated such that the distance between the models' joint coordinates and the estimated joint coordinates is minimised. To calculate the pose of the palm, the position of the wrist and the position of the metacarpal joints of the fingers are needed. Their 3D positions in world coordinate space are obtained from the point cloud by using the surface point lookup for the corresponding keypoints. To get the absolute palm pose, the Kabsch algorithm [102][8] is applied. The output transformation minimises the mean squared distance between all pairs of corresponding points.

The flexion angles of a finger i are calculated as follows: Let $p_0, p_1, p_2, p_3 \in \mathbb{R}^3$ denote the estimated joint coordinates from metacarpal to tip. The normalised bone lengths are given by $b_j = \frac{p_{j+1} - p_j}{\|p_{j+1} - p_j\|} \forall\ j = 0, 1, 2$. Then, $\theta_2(i) = \frac{\pi}{2} - \cos\left(b_0[2]\right)$ where $b_0[2]$ denotes the z-component of b_0. To get the bending of the interphalangeal joints, we project the distal phalanx into the plane spanned by the middle and proximal phalanx. Let $b_2' = (b_2 - \langle b_2, -b_0 \times b_1 \rangle \cdot (-b_0 \times b_1))$ denote the projected vector. Then, $\theta_3(i) = \operatorname{atan2}\left(\left\langle b_0 \times b_2', \frac{-b_0 \times b_1}{\|-b_0 \times b_1\|} \right\rangle, \langle b_0, b_2' \rangle\right)$. Finger spreading is half the angle between the proximal phalanxes of the index and little fingers (both projected into the plane of the palm).

4.5. Pose Refinement

In the previous sections, we segmented hands and obtained an initial estimate for the kinematic model for the hand. In this section, we refine the estimate. The approach we follow uses bounded line search and multiple starting points. The search space is a subset of $\mathbb{R}^{15} \times \mathrm{SO}(3)$ corresponding to all hand poses, where $\mathrm{SO}(3)$ is the space of all rotations in three dimensions. The error function is combined from several components

[8] Implementation follows: `https://web.archive.org/web/20200712141811/https://nghiaho.com/?page_id=671`

Description	Joint	Limit
Flexion of all joints $j \in \{2, 3\}$ of all fingers $i \in \{0, \dots 4\}$	$\theta_j(i)$	$\pi \,\mathrm{rad}\,\mathrm{s}^{-1}$
Translation wrist	-	$2\,\mathrm{m}\,\mathrm{s}^{-1}$
Rotation wrist	-	$\pi \,\mathrm{rad}\,\mathrm{s}^{-1}$
Adduction thumb	$\theta_1(i)$	$\pi \,\mathrm{rad}\,\mathrm{s}^{-1}$
Adduction (spreading) fingers $i \in \{1, \dots 4\}$	$\theta_1(i)$	$0.5\pi \,\mathrm{rad}\,\mathrm{s}^{-1}$

Table 4.4.: Velocity limits of the hand.

Component	Weight
Acceleration	0.1
Distance 2D keypoints	0.6
Skeleton keypoint close to the point cloud	0.2
Distance of the centroid	0.05

Table 4.5.: Weights of all the components that form the error function.

as a weighted sum (Table 4.5). Each component calculates an individual error. Since each component calculates the error in a different space, the error components are scaled before being weighted and added. The—arbitrarily chosen—desired scale is that an error of 1 roughly corresponds to a displacement of all keypoints by $1\,\mathrm{cm}$. The scaling parameters for each of the following components are heuristically chosen to roughly achieve the desired magnitude. The optimisation procedure consists of the following four components:

Acceleration: Calculates a linear motion extrapolation from the previous two wrist poses and forces the current wrist pose towards the calculated one. This ensures a smooth trajectory. The linear extrapolation is calculated as follows. Let $p_{-1}, p_{-2} \in \mathbb{R}^3$ denote the wrist positions and $o_{-1}, o_{-2} \in \mathrm{SO}(3)$ the wrist orientations of the previous two hand poses. Let t_0, t_{-1}, t_{-2} denote the corresponding timestamps when input was received. The extrapolated pose is $\hat{p}_0 = p_{-1} + \Delta t \cdot (p_{-1} - p_{-2})$ with $\Delta t = \dfrac{t_0 - t_{-1}}{t_{-1} - t_{-2}}$. To extrapolate the orientation, we first calculate the orientation difference $\Delta o = o_{-2} \cdot o_{-1}^{-1}$. Then, we transform it into angle-axis representation $\Delta o = (\hat{n}, \alpha) \in \mathrm{SO}(3)$, where $\hat{n} \in \mathbb{R}^3$ is a normalised vector and α the angle, and calculate the extrapolated orientation $\hat{o}_0 = (n, (1 + \Delta t) \cdot \alpha) \cdot o_{-1}$. Values are clamped to meet velocity and joint limits (Table 4.4). Given the wrist position $p_0 \in \mathbb{R}^3$ and orientation $o_0 \in \mathrm{SO}(3)$ of the hand model. The angular difference $\Delta\alpha$ is again calculated by decomposition into axis-angle

Figure 4.6.: Hand skeleton projected into skin segment. The big circle in the middle represents the centroid.

Wrist	0.23664061			
Finger	Metacarpo-p.	Proximal interp.	Distal interp.	Fingertip
Thumb	0.045347154	0.035157289	0.031676931	0.015353241
Index	0.078354179	0.038607894	0.029076354	0.013576274
Middle	0.081827190	0.048613664	0.035887418	0.016265277
Ring	0.078163205	0.042327582	0.028724368	0.012535168
Little	0.066725524	0.028582792	0.024217546	0.012340338

Table 4.6.: Weighting of the keypoints to calculate the centroid. Keypoints forming the palm are given higher weights than the fingers since they span a larger volume. In the table head, phalangeal is abbreviated by p.

decomposition: $\hat{o}_0 \cdot o_0^{-1} = (n, \Delta\alpha)$. The overall acceleration error is a sum of translation and rotation error: $50 \cdot \|p_0 - \hat{p}_0\| + 5 \cdot \Delta\alpha$.

Distance of centroid: Forces the centroid of the hand model towards the centroid of the surface points belonging to that skin segment. We first calculate the centroid of a hand pose $c \in \mathbb{R}^3$ as a weighted sum of the joint coordinates. The weights are given in Table 4.6. Figure 4.6 shows an example. The error is $e = 100 \cdot \|c - c^*\|$ where c^* is the centroid of the point cloud of the skin segment. This completes the description of how the components calculate an error.

Distance 2D keypoints: Forces hand model joints projected into the colour image and keypoints from GANerated hands [99] to match. The purpose is to capture the fine details of finger poses. First, the colour image cropped to the skin segment is passed to the GANerated hands neural network. The output of the neural network is a heatmap on a discrete normalised input square $h_i \colon [0,1]^2 \to [0,1]$ for each joint i. From the heatmap,

Parameter	Value
Probability threshold to accept a keypoint of a heatmap	9 %
Threshold for the average keypoint probability to consider a segment as a hand	20 %
Minimum number of accepted keypoints	5

Table 4.7.: Filter parameters for keypoints

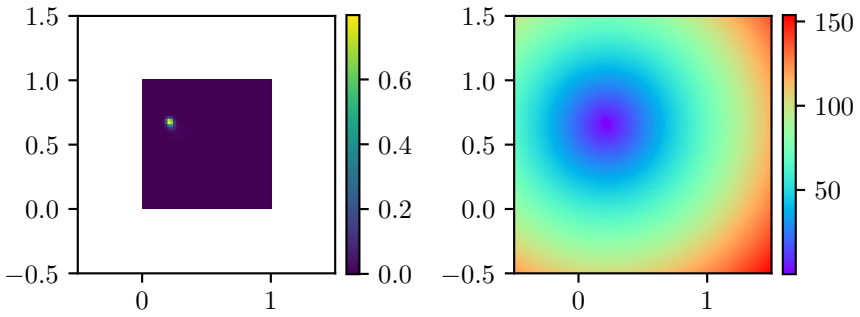


Figure 4.7.: Heatmap with stretching function applied (left) and error function (right) for the tip of the thumb. Note that both plots show areas beyond the extent of the heatmap.

we extract the keypoint, i.e. the location of the peak $p_i^* = \arg\max_{p \in [0,1]^2} h_i(p)$. Since we access the heatmap with real-valued (potentially unbounded) indices, the heatmap is padded with -0.5 outside the skin segment and uses bilinear interpolation for values between integer locations. Keypoints are filtered by the parameters in Table 4.7. Let $p_i \in [0, 1]$ denote the joint i of the hand model projected into the heatmap i. Its error is calculated as follows:

$$e(i) = \begin{cases} \frac{1}{6} \cdot (1 - g\left(h_i(p)\right)) & \text{if } g\left(h_i(p)\right) > 0.4 \\ 2.1 - 5 \cdot g\left(h_i(p)\right) & \text{if } 0.4 \geq g\left(h_i(p)\right) > 0.1 \\ \frac{18}{\text{box width}} \cdot \|p^* - p\| & \text{otherwise} \end{cases} \tag{4.7}$$

where box width refers to the length of the bounding box calculated in Section 4.3. Figure 4.7 shows an exemplarity heatmap on the left and the corresponding error function on the right, assuming a bounding box width of $20\,\mathrm{cm}$. The last condition in Equation 4.7 is responsible for the cone shape of the error function. The first two conditions only play a role if the peak is spans multiple pixels to then generate a broader local minimum. Since the output values of h_i tend to be small, we apply a stretching function $g\colon [0, 1] \to [0, 1]$

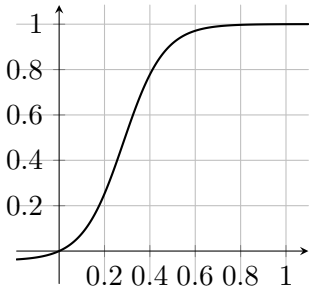


Figure 4.8.: Stretching function $g(x) = 0.52129 \cdot \tanh(5.5876 \cdot x - 1.5824) + 0.47905$ applied to heatmap values.

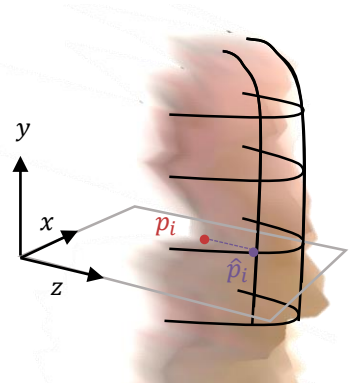
shown in Figure 4.8. We then calculate the average error and discard keypoints from the error function whose error is 1.5 times above the average error (the neural network sometimes generates incorrect keypoints). The returned error value is the average error of the remaining keypoints, with the wrist given a 4-times weight.

Skeleton keypoint close to surface: Forces the joint coordinates to be close to the point cloud but still further away from the camera than the reference point of the point cloud. This aligns the hand pose with the point cloud and counteracts other components that would push the hand or individual fingers outside the skin segment. We first filter keypoints that are occluded. This condition is met if the point cloud has a non-skin coloured point that is closer to the camera and matches the keypoint when projecting it into the image. Keypoints are sorted by proximity to the camera. We then take the five closest keypoints. For each keypoint $p_i \in [0,1]^2$, we obtain the corresponding point of the point cloud $\hat{p}_i \in \mathbb{R}^3$ and normal estimation $n_i \in \mathbb{R}^3$ from the surface point lookup described in Section 4.3 (Figure 4.9a). We then calculate a reference point $p_i^* = p_i - 0.5 r_f \cdot n_i$ (Table 4.8) that is close to the point cloud points and inside the hand

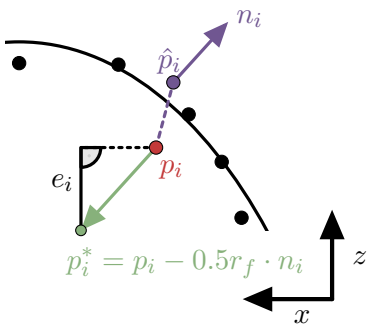
[9] Those small values are picked such that the effects of parameter changes on keypoints in 3D and 2D space can be approximated by a line.

Description	Parameter
Finger radius [103]	$r_f = 4\,\mathrm{mm}$
Minimum improvement of the error function	$\Delta_{\text{min}} = 0.0001$
Maximum number of steps	$n_{\text{max}} = 30$
Trust region size for translational parameters	$1\,\mathrm{cm}$
Trust region size for rotational parameters[9]	$0.2\,\mathrm{rad}$
Number of samples	$n_\alpha = 9$

Table 4.8.: Parameters for the components of the error function and the line search.



(a) Surface mesh of the index finger (Figure 4.2b) approximated by a black, smoothed surface grid. The grey quadrangle represents a slice from the 3D space perpendicular to the y-axis that includes p_i and $\hat{p}_i$.



(b) Slice from the left image. The black points represent points of the point cloud, and the arc represents the surface grid. The surface point $\hat{p}_i$ is the one closest to p_i when projecting the point cloud into the image plane of the camera, but not necessarily the closest one in 3D space. Assuming the camera position o is far above, e_i is the segment length after orthogonal projection of p_i onto p_i^*.

Figure 4.9.: Calculation of the component for the skeleton keypoint close to the surface.

(Figure 4.9b). If the angle between n_i and the z-axis is larger than $45°$, we use $P_c \cdot p_i^*$ instead of the keypoint to get a new surface point, normal, and reference point. The motivation is that laser shadows create surface points lower than the hand and with a normal roughly perpendicular to the z-axis (Figure 4.2b). If a keypoint is outside the skin segment, the surface point lookup gives us a point from these laser shadows, which is likely too low. Using $P_c \cdot p_i^*$ for the lookup returns a surface point further away from the edge of the skin segment and therefore not prone to the laser-shadow problem. The error is then $e_i = \max\{0, \|p_i - o\| - \|p_i^* - o\|\}$ where $o \in \mathbb{R}^3$ refers to the position of the camera, i.e. e_i denotes how much further p_i^* is from the camera than p_i. The overall error is the average multiplied by 100.

The line search proceeds as follows. In each step, we select a dimension (starting from the wrist pose and orientation, followed by the finger joints), and a trust region (Table 4.8) and calculate a step size. Each component calculates an optimal step size within that radius. The overall step size is the weighted sum of the components' best step sizes. Using the step size, we calculate a new hand pose. If its error is larger than the previous candidate, we discard it and continue with the previous candidate. Line search with a small search radius was chosen because the difference of keypoints after projection into image space can be approximated by a line. This eases the calculation of optimal vector lengths to get the optimal step size of the component. To each component,

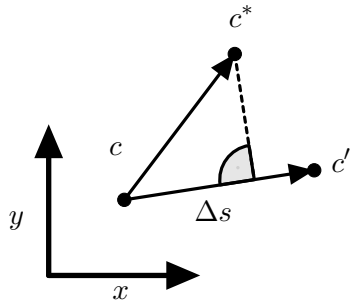


Figure 4.10.: Calculation of the step size Δs for the centroid component. Note that Δs is not the length of the line segment but the fraction of its length compared to c, c'.

we pass the current hand pose and a hand pose at the boundary of the trust region (referred to as the next pose):

Acceleration: Best step size is calculated as the factor by which each component of the difference vector must be multiplied to get the reference pose, clamped to $[-1, 1]$. To find the best orientation, we first calculate the angle α between the current and next pose. Next, we need to find a scaling for the angle such that the orientation is closest to the reference. Sparing an analytical solution, we do this by sampling. We take n_α (Table 4.8) samples l equally distributed along $[-1, 1]$, calculate the new orientation $\alpha' = l \cdot \alpha$ and take the sample l^* as the best step size where $l^* \cdot \alpha$ is closest to the reference orientation.

Distance of centroid: Let $c, c' \in \mathbb{R}^3$ denote the centroid of the current and next hand pose. Let $c^* \in \mathbb{R}^3$ be the centroid of the point cloud of the skin segment. The step size is obtained from the orthogonal projection of $c^* - c$ onto $c' - c$ (Figure 4.10): $\Delta s = \dfrac{\langle c^* - c, c' - c \rangle}{\|c' - c\|^2}$. If the movement is orthogonal to $c^* - c$, i.e. $\cos\alpha < \cos(45°)$ with $\cos\alpha = \left| \left\langle c^* - c, \dfrac{c' - c_i}{\|c' - c\|} \right\rangle \right|$, Δs is multiplied by $\dfrac{\cos\alpha}{\cos(45°)}$. Afterwards, the step size is clamped to $[-1, 1]$.

Distance 2D keypoints: For each keypoint i, the heatmap tells us the optimal position $p_i^* \in [0, 1]^2$ in the 2D image space. Let $p_i \in \mathbb{R}^2$ denote the keypoint of the current pose and $p_i' \in \mathbb{R}^2$ denote the keypoint of the next pose. We calculate the orthogonal projection of p_i^* onto the line (p_i, p_i'): $\Delta s_i \cdot (p_i' - p_i) = \dfrac{\langle p_i^* - p_i, p_i' - p_i \rangle}{\|p_i' - p_i\|^2} \cdot (p_i' - p_i)$. Figure 4.11 illustrates the procedure for the tip of the thumb [10]. Then, Δs_i clamped to $[-1, 1]$ gives

[10] The distance between p_4 and p_4' is exaggerated for better readability

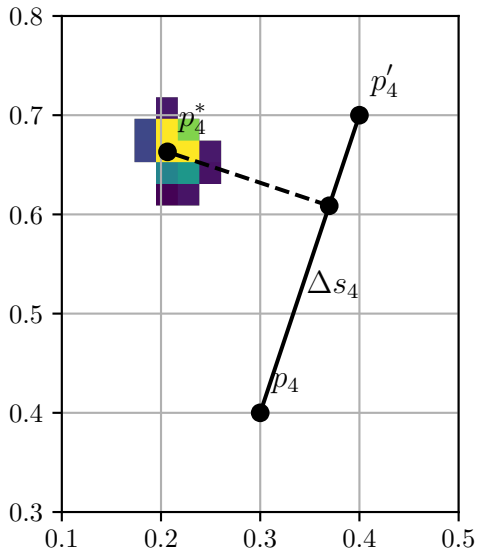


Figure 4.11.: Crop from the heatmap of Figure 4.5 with only the thumb tip ($i = 4$). The lines show how Δs_4 is calculated from the orthogonal projection of p_4^* onto the line p_4, p_4'. Note that Δs_4 is not the length of the line segment but the fraction of its length compared to p_4, p_4'.

the optimal step size with regards to keypoint i. The best step size over all keypoints is the average, with the wrist given a four times higher weight. Keypoints filtered in the error calculation are disregarded here, too.

Skeleton keypoint close to surface: Here, we consider each of the five keypoints $p_i \in \mathbb{R}^3$ included in the error calculation. Let $p_i' \in \mathbb{R}^3$ denote the keypoint of the next pose. We use the reference point $p_i^* \in \mathbb{R}^3$ and error e_i from the error calculation. If the error is small or zero, the point is skipped (i.e. it does not influence the returned step size). The calculation of the step size is similar to Figure 4.10, we just replace $c^* - c$ by $e_i \cdot (0,0,1)^T$ and $c' - c$ by $p_i' - p_i$. We thus scale $p_i' - p_i$ to a length of e_i and take the z-component to get the step size: $\Delta s_i = \frac{e_i}{\|p_i' - p_i\|^2} \cdot \left\langle (0,0,1)^T, (p_i' - p_i) \right\rangle$ and clamp Δs_i to $[-1, 1]$. If $p_i' - p_i$ is orthogonal to z, i.e. $\cos\alpha < \cos(45°)$ with $\cos\alpha = \left| \left\langle (0,0,1)^T, \frac{p_i' - p_i}{\|p_i' - p_i\|} \right\rangle \right|$, Δs_i is multiplied by $\frac{\cos\alpha}{\cos(45°)}$. The overall step size is then the average.

Pose refinement is the last step in the pipeline and can take nearly arbitrary time to reduce the error further. The processing is therefore time-limited. To allow interruption at sensible points, the line search is split into small chunks of work. Each chunk runs three descent steps where the direction of the bounded line search is inverted each time. In the first nine steps, the finger poses are ignored. This ensures that even on a tight

time budget, the position error is significantly reduced. If there is enough time, finger poses are included in later steps. The descent for a hand stops when the improvement drops below Δ_{min} or n_{max} steps have been executed (Table 4.8). The descent stops if the initial hand pose of the next frame arrives once the chunk of work is completed. The hand poses with the lowest error are then added to the hand trajectory, and the pose refinement thread continues with the hand poses of the next frame. If a frame contains multiple hands, new hands or those without a previous pose are prioritised. Afterwards, the hand pose with the lowest number of steps so far is picked next.

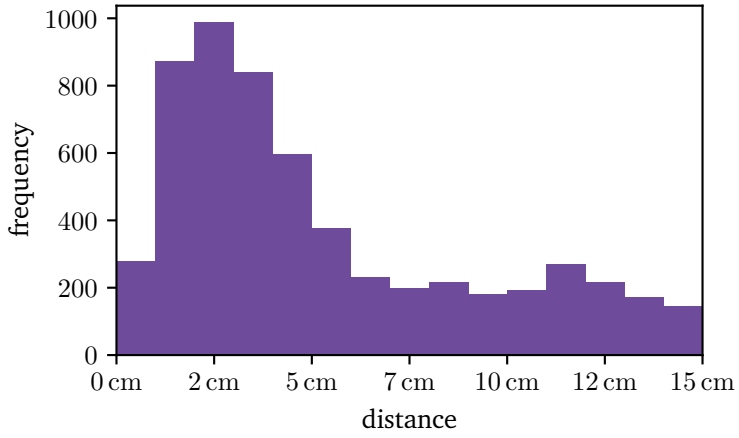
4.6. Evaluation and Sensor Information Fusion

This section presents a small evaluation of the performance of the framework. The evaluation uses 16 trials from the study on non-verbal communication (Section 3.1) where the OptiTrack[11] system delivered continuous hand trajectories for all four hands. Our tracking framework was run with the recorded input from the Kinect 2. We use the wrist position calculated by our framework and use the recorded positions of the OptiTrack markers to calculate the ground truth. A fixed offset is added to the position, and then the closest point of the point cloud is taken. The offset is necessary because the markers are positioned below the wrist and do not record the flexion of the wrist. Associations between hands tracked by the framework and ground truth are formed based on proximity. Sometimes, the tracking could incorrectly identify the tan blocks or forehead as the hand. Since ground truth and tracked hand then differ significantly, associations are deleted when hand certainty drops below $50\,\%$ or the distance exceeds $15\,\text{cm}$.

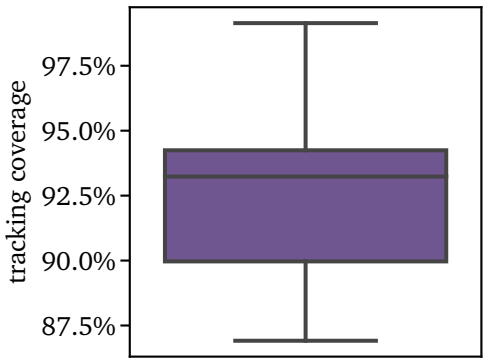
The tracking is run in real-time and fully utilises the hardware of the system ($16\,\text{GB}$ of DDR4 RAM and an Intel i5-8600K CPU with six cores and a maximum speed of $4.1\,\text{GHz}$). The offset between tracked hands and ground truth has a median of $3\,\text{cm}$ to $4\,\text{cm}$ (Figure 4.12a). However, associations must be discarded after $3\,\text{seconds}$ on average (Figure 4.12c). Moreover, for $6.7\,\%$ of the time, no tracked hand is close enough to the ground truth to be counted as matching (Figure 4.12b). This miss rate reaches values up to $13\,\%$ in some trials. We therefore conclude that the framework contributes towards accurate hand pose estimation but is insufficient for reliable, continuous hand tracking.

The core insight from the evaluation is that the framework above struggles with finding new hands. Hands are detected in other skin-like regions or in incorrect orientation. We

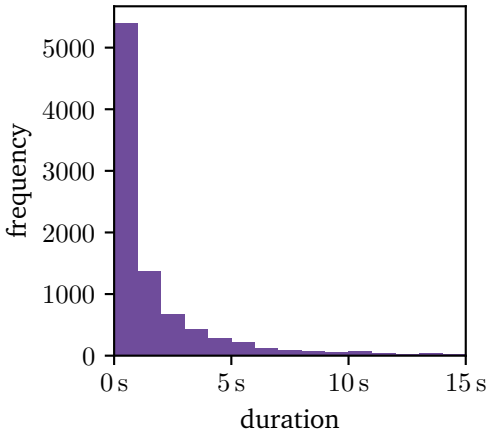
[11] OptiTrack Flex 3: `https://web.archive.org/web/20201029041019/https://optitrack.com/cameras/flex-3/` (date accessed: 2025-02-21)



(a) Histogram of the distance between tracked wrist pose and ground truth wrist pose (from the OptiTrack system) for all hand poses.



(b) Fraction of the duration where hands are successfully tracked.



(c) Length of time segments per hand where that hand is continuously tracked. Higher values indicate better performance.

Figure 4.12.: Evaluation of hand tracking with the setup from Section 3.1.

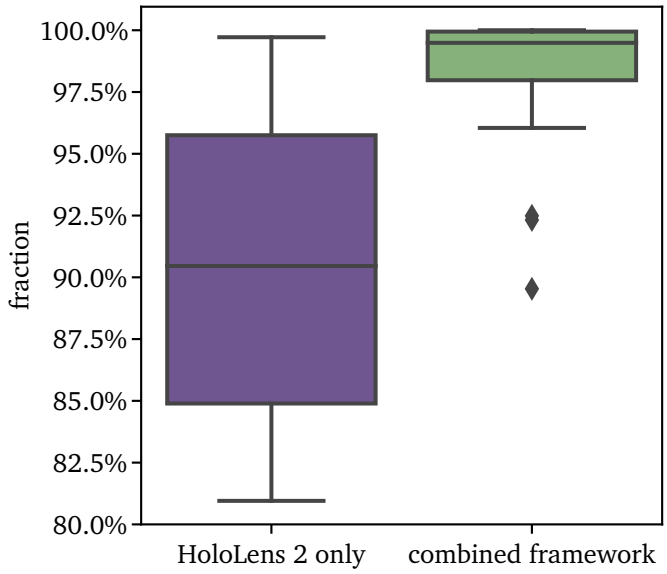


Figure 4.13.: Fraction of the duration where hands are successfully tracked with the setup from Section 7.1. Higher values indicate better performance.

therefore combine the robust hand-tracking capabilities of the HoloLens 2 with the broader sensing range of the top-mounted depth camera. The hardware components and registration process are described in Appendix B.1. The HoloLens 2 calculates the keypoints for both of the user's hands and sends them to the main application at a rate of $60\,\mathrm{Hz}$. There, the inverse kinematics is applied to transform the keypoints into joint angles. This eliminates tracking inconsistencies that would sometimes lead to unnatural hand poses. If no hand position is received for $200\,\mathrm{ms}$, the recognition pipeline from Figure 4.1 is used to close gaps in the hand tracking data.

Another small evaluation shows how the combined framework improves tracking coverage. The evaluation criterion is the percentage of the duration for which we obtain hand poses. The study setup is the same as in Section 7.1 but with shorter task durations. Overall, the evaluation includes 128 hand trajectories from 16 participants executing the assembly task for $175\,\mathrm{s}$ to $541\,\mathrm{s}$. Hand trajectories where the HoloLens 2 tracked the hand for less than $80\,\%$ of the time are filtered. Below this threshold, participants rested their hand below the table for too long. This results in 52 trajectories remaining. From those, the HoloLens 2 tracked the hand for $90.4\,\%$ of the time (Figure 4.13). Our combined

framework fills the gap and achieves a coverage of $98.7\,\%$. However, the framework only achieves a refresh rate of $2\,\mathrm{Hz}$ due to the overall load of the software[12].

4.7. Conclusions and Discussion

This chapter discussed readily available frameworks for hand tracking (Section 4.1). From those, the HoloLens 2, GANerated Hands [99] and the hand tracker from OpenPose [101] are picked. The HoloLens 2 offers accurate and reliable hand tracking but suffers from gaps in the trajectory when the hand leaves the view of the camera. We therefore incorporate a fallback method. The fallback method implements a pipeline of skin segmentation, bounding box tracking, pose estimation, and pose refinement (Section 4.2). Skin segmentation and pose estimation use publicly available algorithms. Novel contribution is the kinematic hand model (Section 4.4), where geometric constraints are achieved by limiting joint angles. Still, the model captures a wide range of dexterity. We present the formulas to convert the estimated joint coordinates and 2D keypoints of a hand into the kinematic model. Pose refinement (Section 4.5) performs a bounded line search in the kinematic model space. Error function is a combination of criteria taking skeleton keypoints from the RGB image, the depth data, and smoothness constraints into consideration. That way, multiple data sources can be integrated. All pipeline steps are presented with formulas and parameters to allow the framework to be reproduced. In combination with the HoloLens 2, an excellent tracking robustness is achieved (Section 4.6). The obtained hand trajectories narrow down the search space for executed actions in the next chapter.

The limitation of the approach is the fine-tuning required for the fallback method. The skin segmentation may need re-training if skin colour or lighting conditions deviate too much from the original training data. Moreover, training data must be manually obtained and labelled. Bounding box tracking is tuned towards a camera mounted at the ceiling at a distance of $1.5\,\mathrm{m}$. Other setups thus require adjusting parameters. The pipeline is computationally very expensive and therefore bottlenecked by the CPU processing capabilities. Overall, solutions for 3D hand tracking are still evolving alongside head-mounted devices. In the future, robust and accurate hand tracking might therefore be available out of the box.

[12] Each obtained hand pose thus accounted for up to $500\,\mathrm{ms}$ of tracking duration if no other tacking point fell within this period.

CHAPTER 5

Task State Modelling and Tracking

The previous chapter introduced robust hand tracking to support tracking the human's actions. This chapter lays the foundations so that the robot knows prior and future actions. How these actions are executed is irrelevant to this chapter since we address the details of action execution in Section 7.1. This chapter focuses on representing the current state and location of all objects in a formal model that enables efficient updates and facilitates reasoning about future actions.

The remainder of this chapter is split into six parts. We first review related literature on object detection, object classification, and action detection. Next, we sketch our approach to object detection (Section 5.2). The main part of this chapter is the formalisation of the

task model and how the state of the task is tracked with it (Section 5.3 and Section 5.4). We then present an algorithm for selecting reasonable (later referred to as goal-oriented) actions from all currently executable actions (Section 5.5). Finally, we evaluate our approach in simulation (Section 5.7).

5.1. Related Work

Monitoring both objects in the robot's workspace and human actions is essential for effective human-robot collaboration. This is typically split into the components of human pose estimation and object identification [104, 72]. The former one ranges from simple human position to full human pose estimation. The process of human pose estimation in isolation is well-studied [105], and ready-to-use frameworks exist (e.g. [77, 75, 76]). All of them are targeted towards inputting image sequences, optionally with depth data. However, using these frameworks in collaboration scenarios still faces many challenges:

- Most frameworks rely on keypoint detection for single images. If that fails, this can result in abrupt pose changes or a breakdown of the tracking.
- The human leaves the area of perception. Tracking cannot then proceed, and an initial pose estimation needs to be established.
- Parts of the human are occluded, e.g. by the robot arm. Essential actions might not be observed, and the recorded action sequence remains incomplete.

The research on robust activity detection of industrial tasks suited for collaboration scenarios is still in an early stage (e.g. [106, 107, 108, 109, 40]). Human activity tracking as a key element for task state tracking remains—at the moment—largely impractical. We therefore focus on object identification for task state tracking. Objects in the scene—unless manipulated by an agent—remain stationary, making them easier to detect than actions. Consequently, a few robust observations are sufficient to determine the current state of the workbench.

As with activity detection, object detection has been well-studied in isolation [110, 111, 112], but significant challenges remain for its application in the scenario under consideration. The primary challenges are: (i) to achieve robust object detection under varying light conditions [113], (ii) to recover from false recognitions [113], and (iii) to explicitly model occlusion [72]. We assume that in the future, robust and ready-available systems for identification of known, industrial components exist (see e.g. [114, 115, 116, 117] for preliminary approaches). These systems may rely on primitive feature detections

or employ neural network architectures. In both cases, the result of the classification is some probability for each object class. Our approach is therefore designed to work with the output of either of these systems. An important factor when choosing the object detection algorithm is to obtain explicit occlusion information. Modern approaches often leverage convolutional neural networks for object detection and achieve high accuracies [72]. However, occlusion detection with neural networks is still in the early stages (e.g. [118]).

We therefore pursue an approach that uses hand-crafted features, which allows us to extract occlusion information. Hand-crafted features are still relevant (e.g. [119, 120]) when it would be too tedious to acquire sufficient training data. Another benefit is that information from the workspace layout, object CAD data, depth, and occlusion information can be incorporated more easily. In the remainder of this thesis, we restrict object categories to simple-shaped monochrome objects (referred to as *blocks*), as they can be found in building block collections for children. These are motivated by the reference task designed for the user study (Section 7.3).

We follow the established pipeline (see e.g. [121, 122, 119, 120]) to obtain coloured point clouds from RGB-D cameras and compare object candidates with known CAD models. Features to compare with are object size, shape, and colour. The incorporation of occlusion leads to additional challenges when tracking the task state. Approaches in the literature use the pose of known foreground objects to model occlusion (e.g. [17, 123]). These approaches often have difficulties with moving or deformable objects, as is the case for the human arm, hand, and robot. We therefore introduce a lightweight, heuristic approach to model occlusion.

When it comes to object tracking under occlusion for human-robot teaming, the literature uses *ageing*-based approaches, which run object detection for each frame and increase or decrease the presence likelihood (e.g. [53, 124]). Baraglia et al. [53] remove and add objects based on some likelihood threshold. The main drawback of this approach is its lack of guaranteed consistency, which can result in frequent false disappearances and the misdetection of objects. The latter can even lead to an overestimation of the number of present objects. Riedelbauch, Werner and Henrich [124] use ageing as a measure of uncertainty that the object has been manipulated. Their task state updates require that each action is fixed in its pick and place location and object type. The framework can thus not track pick-and-place actions where multiple sources and targets exist per object type. Our approach extends towards mutually exclusive or partially observable actions.

Description	Parameter
Length of laser-shadows	$d_l = 2\,\mathrm{cm}$
Difference in colour value	$\delta_c = 100$
Minimum dimensions of an object	$\delta_z = 4\,\mathrm{mm}$
Height of the structure	$h_s = 10\,\mathrm{cm}$
Minimum height of the hand above the structure	$h_h = 12\,\mathrm{cm}$
Minimum number of points to form an object	$n_{\text{min}} = 6$
Standard deviation of depth noise along the z-axis	$\sigma_d = 4\,\mathrm{mm}$
Distance threshold for matching a point	$t_d = 1\,\mathrm{mm}$
Lower threshold for object similarity	$t_{ls} = 35\,\%$
Threshold for occlusion (percentage of points)	$t_o = 10\,\%$
Threshold for presence	$t_p = 50\,\%$

Table 5.1.: Parameters for object detection ordered by parameter identifier. Most of the parameters are heuristically chosen when coding and testing the object detection.

5.2. Object Detection and Classification

The overall setup consists of a robot, a workspace layout, and a top-mounted depth camera. The workspace layout (e.g. Figure B.2), as introduced by Riedelbauch [17],is a large piece of paper on which the positions of objects are printed. By design, it is aligned with the robot's workspace coordinate system, easing the specification of positions for the robot. The depth camera is registered to the workspace layout such that positions in the viewport of the camera can be projected into the workspace layout with sub-centimetre accuracy (Appendix B.1). Object shapes and orientation are assumed to be known and fixed for each position in the workspace layout. Thus, axis-aligned bounding boxes can be placed in the camera's coordinate system at each position where objects are to be expected. We refer to these bounding boxes as *places* in reference to the task model based on Petri nets (Section 5.3).

Let $P_C \subset \mathbb{R}^3$ denote a point cloud with a function $c_{P_C} \colon P_C \to [0, \ldots, 255]^3$ assigning each point an RGB colour value. We assume that the point cloud P_C has already been transformed from the camera into the workspace coordinate system and the surface of the workspace has been removed, i.e. all points with a z-value below σ_d (Table 5.1). For a given point $p \in \mathbb{R}^3$, we use the shorthand notation $p[i]$ with $i \in \{0, 1, 2\}$ to refer to its i-th component, e.g. $p[2]$ is the z-component of p.

We first present the **classification** of an object since the result of the classification procedure is used as part of the detection algorithm. The classification makes the fol-

lowing three assumptions: (i) Objects are enclosed in bounding boxes at known places. (ii) Objects are monochrome. (iii) A database with the reference objects exists.

Let P_B denote the intersection of P_C with some bounding box B, i.e. O is an object represented within P_C. In the following, we calculate a tight axis-aligned bounding box around P_B and denote its diagonal by $d_b \in \mathbb{R}^3$ and its centre by $t_b \in \mathbb{R}^3$. Moreover, let $R = (d_r, \text{col}_r, M_r)$ denote a reference object where $d_r \in \mathbb{R}^3$ is the diagonal vector of its bounding box, $\text{col}_r \in [0, \ldots, 255]^3$ the mean colour, and $M_r \subset \mathbb{R}^3$ the surface as an infinite set of points—for the implementation, this can be represented by a polygon mesh.

As in Section 4.3, we use $\text{bell}(x, \sigma) = e^{-\frac{x^2}{2\sigma^2}}$ to denote the bell curve. The bell curve transforms the error measure for size, colour, and shape into a similarity value in the range $[0, 1]$. The size similarity is based on the differences in edge lengths of the bounding boxes. The standard deviation σ is set to the minimum object height δ_z:

$$\text{sim}_{\text{size}}(P_B, R) = \sqrt[3]{\prod_{i \in \{0,1,2\}} \text{bell}(d_b[i] - d_r[i], \delta_z)}. \tag{5.1}$$

The colour similarity simply uses the distance in sRGB space of the mean object colour and reference colour. The standard deviation σ is set to roughly a third of the edge length of the cube forming the colour space (δ_c). The term $|\cdot|$ denotes the cardinality of a set:

$$\text{sim}_{\text{colour}}(P_B, R) = \frac{1}{|P_B|} \cdot \sum_{p \in P_B} \sqrt[3]{\prod_{i \in \{0,1,2\}} \text{bell}(c_{P_C}(p)[3] - \text{col}_r[i], \delta_c)}. \tag{5.2}$$

The shape similarity calculates the distances between each point of the point cloud and the surface of the reference object M_r:

$$\text{sim}_{\text{shape}}(P_B, R) = \frac{1}{|P_B|} \cdot \sum_{p \in P_B} \text{bell}\left(\inf_{m \in M_r} \|p - t_b - m\|, \frac{\delta_z}{2}\right). \tag{5.3}$$

Finally, the overall similarity is the geometric mean of Equations 5.1 to 5.3. The geometric mean—in contrast to the arithmetic mean—gives higher weight to the similarity measure that deviates more strongly from one:

$$\text{sim}_{\text{object}}(O, R) = \sqrt[3]{\text{sim}_{\text{size}}(O, R) \cdot \text{sim}_{\text{colour}}(O, R) \cdot \text{sim}_{\text{shape}}(O, R)}. \tag{5.4}$$

The process for the **detection** of objects iterates over all places B and categorises them as *empty*, *occluded*, or object *present*. The result of the categorisation process is formalised

Algorithm 1: Detection of a single object in the workspace in a single frame

input : Given a point cloud $P_C \subset \mathbb{R}^3$, an axis-aligned bounding box B and a matrix $M \in \mathbb{R}^{3\times4}$ to project points onto the flat surface of the workspace.
output : OCCLUDED, PRESENT, EMPTY

Turn B into a point cloud $P_B \subset \mathbb{R}^3$ by sampling the faces that point towards the camera.
$\text{counter}_{\text{present}} = \text{counter}_{\text{occluded}} = \text{counter}_{\text{missing}} = 0$
foreach $p_B \in P_B$ **do**
    $p'_B \leftarrow M p_B$
    $p'_C \leftarrow \arg\min_{p \in P_C} \left\| M p - p'_B \right\|$ // A quadtree from PCL is used to efficiently get the result.
    if $\left\| p'_B - p'_C \right\| > t_d$ **then** // No matching point in point cloud found.
        $\text{counter}_{\text{missing}} \leftarrow \text{counter}_{\text{missing}} + 1$
        continue
    if $p_B[2] < p_C[2] - h_s$ **then** // p_C more than h_s above p_B.
        $\text{counter}_{\text{occluded}} \leftarrow \text{counter}_{\text{occluded}} + 1$
    else if $p_B[2] > p_C[2] + h_s$ **then**
        $\text{counter}_{\text{missing}} \leftarrow \text{counter}_{\text{missing}} + 1$
    else
        $\text{counter}_{\text{present}} \leftarrow \text{counter}_{\text{present}} + 1$
        foreach $p \in P_C$ **do**
            if $\left\| p'_B - Mp \right\| < d_l$ *and* $p[2] > h_h$ *and* $p[2] > h_s$ **then** // Uses quadtree to only iterate neighbouring points within d_l
                $\text{counter}_{\text{occluded}} \leftarrow \text{counter}_{\text{occluded}} + 1$
                $\text{counter}_{\text{present}} \leftarrow \text{counter}_{\text{present}} - 1$
                break
if $\frac{\textit{counter}_{\textit{occluded}}}{|P_C|} > t_o$ **then**
    return *OCCLUDED*
$n_O = |P_C \cap B|$ // Number of points constituting the object.
if $n_O < n_{min}$ **then**
    return *EMPTY*
if $\frac{\textit{counter}_{\textit{present}}}{|P_C|} > t_p$ *and* $n_O > 2 \cdot n_{min}$ **then**
    return *PRESENT*
return *EMPTY*

and built upon in Section 5.4.1. Algorithm 1 performs this categorization for a single frame, given the point cloud P_C, the axis-aligned bounding box B, and a projection matrix M into the surface of the workspace[1]. Initially, all places are categorised as empty before starting the object detection. If the categorization of B changes from empty to present—ignoring categorizations of occluded in between—we calculate the similarity $s = \text{sim}_{\text{object}}(P_C \cap B, R)$ where R denotes the reference object associated with B. If s is less than t_{ls}, the categorization of empty for B is kept. This rule reduces the number of incorrect detections when one of the agents is within a place.

5.3. Task Model

The previous section handled the problem of object recognition of single objects in single frames. Now, we elaborate on how to track the overall task state. The description of the task model follows the description in [104]. However, some aspects are skipped[2] or simplified[3] if they are not relevant for the following sections. Moreover, an extension for unstacking objects is included, which has been added after publishing the paper.

Definition 5.3.1. *A* coloured Petri net *is a tuple* (P, T, T_c, O, A, f) *with*

- *a finite set of places* $P = \left\{p_1, \ldots, p_{|P|}\right\}$,
- *a finite set of transitions* $T = \left\{t_1, \ldots, t_{|T|}\right\}$,
- *a finite set of controllable transitions* $T_c \subseteq T$,
- *a finite set of token types* $O = \left\{o_1, \ldots, o_{|O|}\right\}$,
- *a set of arcs* $A \subseteq (P \times T) \cup (T \times P)$, *and*
- *a filter function* $f : A \to O$

where P, T, *and* O *are pairwise disjoint.*

Table 5.2 provides an overview of the intuitive semantics of the components of the Petri net. An illustrative example follows later in this section. Before that some shorthand notation is introduced.

[1] Since our workspace surface is aligned with the x-y-plane, the projection M just discards the z-component

[2] The second filtering step for the feasible transitions is skipped here as well as the mathematical assumptions.

[3] The detection of colours was discarded and replaced by fixed places with one specific type of object. This makes the program more robust. In consequence the model for emissions is simplified.

Component	Semantic
places P	locations of the workspace
transitions T	actions of the human or the robot
controllable transitions T_c	actions only executable by the robot (relevant for Section 5.5)
uncontrollable transitions $(T \setminus T_c)$	actions only executable by the human (relevant for Section 5.4.2)
token types O	types of objects
instance $(p, o) \in I$	which object o is at which location p in the workspace

Table 5.2.: Semantic meaning of the components of the Petri net.

Most places represent a location in the workspace, i.e. an axis-aligned bounding box. The set of arcs A associates input and output places with a transition. The filter function f determines which token types are consumed or produced at which place. We use the shorthand notation $p \xrightarrow{c} t$ to denote $(p, t) \in A$ and $f((p, t)) = c$.

Based on the set of places P and the set of token types O, we define the set of *instances* $I = P \times O$. An instance $i \in I$ can be the input and output of a transition $t \in T$. If it consumes and produces the same token type o at the place p, we refer to i by the term *side condition*. We define the following functions to ease notation[4]:

Side conditions $\text{side}\colon T \to 2^I, t \mapsto \left\{(p, o) \in I \;\middle|\; p \xrightarrow{o} t \text{ and } t \xrightarrow{o} p\right\}$

Pure input instances $\text{in}\colon T \to 2^I, t \mapsto \left\{(p, o) \in I \setminus \text{side}(t) \;\middle|\; p \xrightarrow{o} t\right\}$

Pure output instances $\text{out}\colon T \to 2^I, t \mapsto \left\{(p, o) \in I \setminus \text{side}(t) \;\middle|\; t \xrightarrow{o} p\right\}$

Likewise, we define the set of side, pure input, and pure output places of t. The *marking* M of a Petri net is a relation $M \subseteq P \times O$. A marking represents the distribution of tokens over places. We enforce an upper limit of one token per place. By M_0, we denote the initial marking of the Petri net, and by $\mathcal{M}$, we denote the set of all markings. A transition $t \in T$ is *enabled* with respect to a marking M, if and only if $\text{in}(t) \subseteq M$, $\text{side}(t) \subseteq M$, and $\text{out}(t) \cap M = \emptyset$. The last condition follows from the token limit. An enabled transition can fire. Then it produces a new marking M' where pure input instances are removed and pure output instances generated, i.e. $M' = (M \setminus \text{in}(t)) \cup \text{out}(t)$. By considering sequences of transitions, we extend the concept to reachability of markings. Given two markings M' and M'', M'' is *reachable* from M' if and only if there is a

[4]We use the adjective 'pure' here to distinguish these definitions from the differing ones in [104].

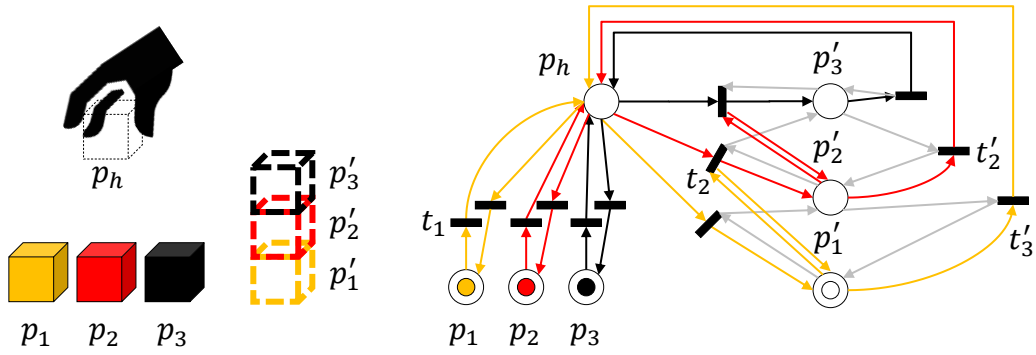


Figure 5.1.: Example for a Petri net and marking. The left depicts the workspace where the three cubes should be stacked as a tower. The right shows the equivalent Petri net including the transitions for construction and dismantling. Big white circles denote places, black rectangles transitions, and arrows arcs. The colour of the arrow indicates the token type, where the grey arrow represents the empty token present in p'_1. The transitions on the right of the stack represent pick actions, the ones on the left represent place actions.

sequence of transitions $t_0, \ldots, t_m \in T$ that when fired starting in M' result in M''. For the implementation, markings are represented as ordered sets of place-token pairs. For each set, a deterministic hash is calculated and stored, so that markings can be checked for equality in constant time. This is relevant in Section 5.4.2, where many equality checks are needed.

A coloured Petri net forms the basic structure to define the task and perform planning operations for the robot. We give a small example of how the Petri net is used to model pick-and-place tasks. Consider the task of stacking three cubes of the colours gold, red, and black as an abstraction of an assembly task. Initially, the cubes are located at positions p_1, p_2, p_3 and should be stacked at p'_1, p'_2, p'_3 as depicted in the left part of Figure 5.1. The task model is constructed as follows: The positions p_1 to p'_3 are used as places. In addition, two places, p_h and p_r, are added for the human and robot (for clarity, the robot is omitted in Figure 5.1). Each cube type gets a token type ●, ●, and ●. For each pick and each place action (for human and robot), a transition is created. Let t_1 denote the transition where the human picks the golden cube from p_1. This requires the arcs $p_1 \xrightarrow{●} t_1$ and $t_1 \xrightarrow{●} p_h$. Let t_2 denote the transition in which the human stacks the red cube on top of the golden one. This requires the arcs $p_h \xrightarrow{●} t_2$, $t_2 \xrightarrow{●} p'_2$, $p'_1 \xrightarrow{●} t_2$, and $t_2 \xrightarrow{●} p'_1$. The last two arcs form a side condition to ensure that the golden cube is in place. For each action, a transition is added to reverse the action. For t_1, we just need to invert the directions of the arcs. When it comes to stacking, we want to ensure that the model does not allow the red cube to be picked at p'_2 while the black is stacked on top at

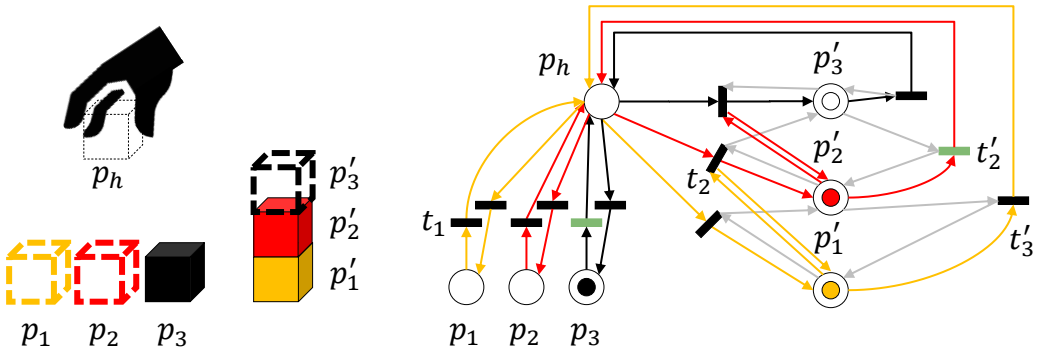


Figure 5.2.: Example from Figure 5.1 in a later state. The golden and red cubes have already been stacked, and the marking of the Petri net has been updated. Green transitions are active.

p'_3. To achieve that, we use the *empty token* ○ to mark the empty place above an object. For transition t_2, we therefore add $p'_2 \xrightarrow{○} t_2$ and $t_2 \xrightarrow{○} p'_3$ to push the empty token to the next place above. Reversing the pick action by transition t'_2 is then achieved by $p'_2 \xrightarrow{●} t_2$, $t_2 \xrightarrow{●} p_h$, $p'_3 \xrightarrow{○} t_2$, and $t_2 \xrightarrow{○} p'_2$. The benefit of the empty token is apparent in Figure 5.2. It ensures that t'_2 is active but not t'_3. Without it, both would be active, although it is not possible to pick the golden cube.

The motivation to include reversal operations is for two reasons:

1. to model when the human performs an undo to fix something, and
2. when a detection error makes the robot think that an action has been executed. In the next step, the detection delivers the correct result, which is that there is no object at the target position.

In both cases, we want the update algorithm not to get stuck. The inclusion of reversal operations, however, increases the complexity of planning. The robot cannot simply pick an active controllable transition. If it were to do that, it might reverse a completed action or—in case more resources are provided than needed—pick an object that is no longer needed. We therefore add more components to the Petri net in Section 5.5 to facilitate goal-oriented planning.

5.4. Task State Tracking

Task State Tracking refers to the process of updating a probability distribution over markings of the Petri net given the sequence of the observed objects over time. Because objects

are only partially observable, we cannot always derive a single marking but sometimes need to keep track of multiple weighted markings, hence the probability distribution. We first formalise the input obtained from the object detection (Section 5.2), referred to as the observation model (Section 5.4.1). Second, we show how to update the belief marking by a Monte Carlo sampling process with pre-filtered transitions (Section 5.4.2).

5.4.1. Observation Model

An *emission* is a tuple $E = \left(E^-, E^0, E^+\right)$ where $E^- \subseteq P$ is the set of empty, $E^0 \subseteq P$ is the set of unobserved, and $E^+ \subseteq P$ is the set of observed places. We enforce that the sets form a partition of P: $E^- \cup E^0 \cup E^+ = P$ and E^-, E^0, E^+ are pairwise disjoint. For observed places, we can derive the token type since all observable places on the workbench can only contain one specific token type. Places with multiple token types, such as the human hand and robot gripper, are never observable. Consider the example in Figure 5.2. There we observe $E^- = \{p_1, p_2, p_3'\}$, $E^0 = \{p_h, p_1'\}$, and $E^+ = \{p_2', p_3\}$. A marking $M \in \mathcal{M}$ is *valid* with respect to an observation E, if and only if (1) for all instances (p, o) in M the place p is in $E^+ \cup E^0$ and (2) for all places p in E^+ there exists a token type $o \in O$ such that $(p, o) \in M$.

Over time, a sequence of emissions $E_1, \ldots, E_n$ is captured. The formal problem is to derive the current probability distribution over markings from the sequence. To simplify notation, we introduce—in analogy to capturing partial observability in Markovian Decision Processes [125]—the *belief marking* $\mathcal{B}_n \colon \mathcal{M} \to [0, 1]$ where $n \in \mathbb{N}_0$ refers to the time index. We define $\mathcal{B}_n$ for some $M \in \mathcal{M}$ as follows:

$$\mathcal{B}_n(M) = \mathcal{P}\left(M \mid E_1, \ldots, E_n\right). \tag{5.5}$$

Specifically for $n = 0$ with the initial, given marking M_0 of the Petri net, we have:

$$\mathcal{B}_0(M) = \mathcal{P}(M) = \begin{cases} 1 & \text{if } M = M_0 \\ 0 & \text{otherwise} \end{cases}. \tag{5.6}$$

Applying that the task state tracking fulfils the Markov property, we derive an inductive calculation procedure for $\mathcal{B}_n$:

$$\mathcal{B}_n(M_n) = \sum_{M_{n-1} \in \mathcal{M}} \mathcal{P}\left(M_n \mid M_{n-1}, E_n\right) \cdot \mathcal{P}\left(M_{n-1} \mid E_1, \ldots, E_{n-1}\right) \tag{5.7}$$

$$= \sum_{M_{n-1} \in \mathcal{M}} \mathcal{P}\left(M_n \mid M_{n-1}, E_n\right) \cdot \mathcal{B}_{n-1}(M_{n-1}). \tag{5.8}$$

where

$$\mathcal{P}\left(M_n \mid M_{n-1}, E_n\right) = \frac{\mathcal{P}\left(E_n \mid M_n, M_{n-1}\right) \cdot \mathcal{P}\left(M_n \mid M_{n-1}\right)}{\mathcal{P}\left(E_n \mid M_{n-1}\right)} \tag{5.9}$$

is defined by Bayes' Theorem. Since the emission E_n only depends on the current marking M_n, we can simplify $\mathcal{P}\left(E_n \mid M_n, M_{n-1}\right) = \mathcal{P}\left(E_n \mid M_n\right)$ and $\mathcal{P}\left(E_n \mid M_{n-1}\right) = \mathcal{P}\left(E_n\right)$. We do not assume a bias towards some emissions. Hence, $\mathcal{P}\left(E_n\right)$ is uniformly distributed and acts as a normalisation factor. We choose an uninformative model for $\mathcal{P}\left(M_n \mid M_{n-1}\right)$ which is only based on the structure of the Petri net. With the helper function

$$\hat{p}_n\left(M, M'\right) = \begin{cases} 1 & \text{if } M \text{ is reachable from } M' \text{ and } M \text{ is valid w.r.t. } E_n \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

we simplify Equation 5.8 to:

$$\mathcal{B}_n(M) = \frac{1}{\alpha} \cdot \sum_{M' \in \mathcal{M}} \hat{p}_n\left(M, M'\right) \cdot \mathcal{B}_{n-1}(M'). \tag{5.11}$$

The normalisation factor is given by

$$\alpha = \sum_{M \in \mathcal{M}} \sum_{M' \in \mathcal{M}} \hat{p}_n\left(M, M'\right) \cdot \mathcal{B}_{n-1}(M'). \tag{5.12}$$

5.4.2. Update Procedure

In the next step, we show how to represent $\mathcal{B}_n(M)$ and how to efficiently evaluate Equation 5.11. We represent $\mathcal{B}_n(M)$ by a dictionary where the keys are markings and the values are the probabilities. The markings with zero probability are not stored in the dictionary. The efficient evaluation of M in Equation 5.11 is achieved in three steps.

The first—preparatory—step consists of firing controllable transitions $t \in T_c$ if an action has been completed by the robot. This is done for all markings M in $\mathcal{B}_{n-1}$. If t is not active in M, then M is discarded. This generates a new dictionary $\mathcal{B}'_{n-1}(M)$.

The second step prepares for the firing of uncontrollable transitions. For that, we exclude all transitions that cannot have been fired. These are, e.g., pick actions where the object still resides at the initial position. This second step constructs the set of *non-blocked transitions* $T_\Theta \subseteq T \setminus T_c$. A detailed description of the second step is provided in previous work [104]. The detailed version adds more steps to further reduce T_Θ.

Description	Parameter
Distance threshold for hand centroid	$d_h = 6\,\mathrm{cm}$
Duration threshold for hand	$\Delta_h = 5\,\mathrm{s}$
Repetitions of the Monte Carlo sampling process	$N = 2000$

Table 5.3.: Parameters for the update procedure. Parameters are heuristically chosen when coding and testing the update procedure.

Here, we briefly sketch the simplified version of the algorithm. Let E_n denote the current and E_{n-1} the previous emission. We define the set of unchanged places $P_u = \left(E^-_{n-1} \cap E^-_n\right) \cup \left(E^+_{n-1} \cap E^+_n\right)$. A transition $\mathbf{t}$ is in T_Θ when it meets all of the following criteria:

- Side conditions must not be empty: $\forall (p, o) \in \mathrm{side}(\mathbf{t})\colon (p, o) \notin E^-_{n-1} \cap E^-_n$
- No arc leads to an unchanged place: $\forall (p, o) \in \mathrm{out}(\mathbf{t})\colon p \notin P_u$
- No arc originates from an unchanged place: $\forall (p, o) \in \mathrm{in}(\mathbf{t})\colon p \notin P_u$
- If one of the pure input or output places (excluding side conditions) p of $\mathbf{t}$ represents a bounding box B in the workspace, the user's hand centroid must have been at most d_h away from the centre of B up to Δ_h in the past (Table 5.3).

The last criterion eliminates transitions associated with places that the user has not manipulated but still fulfil the other criteria (e.g. due to occlusion). We form another subset T^+_Θ of T_Θ to guide the sampling process in the third step. The set T^+_Θ contains all transitions where we observe a new output. That is, there is a pure output place p of $t \in T^+_\Theta$ such that $p \in E^+_n$ but $p \notin E^+_{n-1}$.

The third step represents the core component of the updating procedure. It probabilistically generates new markings in a Monte-Carlo-like fashion. We start by sampling a marking M from $\mathcal{B}_{n-1}$. Afterwards, we fire transitions from T_Θ until we reach a valid marking M', or there are no enabled transitions left in T_Θ. If M' is valid, we increment a counter associated with M' and continue with the next sample. We repeat the sampling process N times. In the end, we divide the counters for each marking M' by the total sum of counters[5] to get an approximation for the probability distribution $\mathcal{B}_n$. This heuristic approach promises to be much faster than an analytical solution that would need to enumerate all reachable markings. We note that one must be careful when choosing a transition from T_Θ:

[5] The total sum of counters is smaller than N, if samples did not produce a valid marking.

1. The Petri net can contain loops. To avoid infinite looping, each transition in T_Θ is fired at most once per sample run.

2. If we have multiple agent places and sources for pick locations, then several markings are consistent with the latest emission. To keep the number of markings small, we prefer transitions from T_Θ^+. Transitions from $T_\Theta \setminus T_\Theta^+$ are considered only when T_Θ^+ contains no active, unused transition.

There are situations where the sampling process does not produce any valid marking. In that case, we rerun the sampling but construct T_Θ without the last criterion because it might have been too restrictive. If the second run does not produce a valid marking either, then M is kept. In that case, the failure to produce a valid marking is attributed to contradicting sensor observations due to noise. For subsequent updates, we continue to use M and E_{n-1} as the previous emission. Thanks to the capability of the algorithm to deduce unobserved actions, even longer action sequences can be recovered, e.g. when half of the task is already completed. The Petri net is not limited to tracking the current state; it can also be used to plan the robot's next action.

5.5. Action Planning

We first need to define the goal of the task. We add a dedicated place as the goal and refer to it as the *goal place*. We make one assumption to simplify the following planning algorithm: All intermediate steps of the task produce a pure output instance that still exists when the task is completed. In the following, we refer to these pure output instances as *goal instances*. In the context of pick-and-place tasks, that means we are not allowed to place something, e.g. a supportive structure, and later remove it before completing the task. With this assumption, we create the *goal transition* t_g. The transition t_g produces a unique token at the goal place, indicating the termination of the task. It has all the goal instances as side conditions. When running the update procedure, t_g is explicitly checked and fired when activated. Other components can then check whether the goal place is marked.

Constructing t_g has the benefit that we can store the goal instances and goal completion in a standard way in the Petri net. Another benefit is that we can use the Petri net to track a sequence of tasks. To do that, places and transitions are created to indicate the completion of each task. One of these places is set as the goal place. After the goal place has been marked and subsequent software components have been informed about the

task completion, the marking from the goal place is erased, and the next task goal is set as the goal place. That way of modelling sequences of tasks is used in Section 7.3.

Reversible actions and exclusive alternatives pose the challenge for the system to find the next action to execute in order to contribute to the task goal. Since pick and place actions are modelled by separate transitions, we need to find a corresponding place action before executing a pick action with Algorithm 2.

Algorithm 2: Goal-oriented transitions

input : Given a coloured Petri net $N = (P, T, T_c, O, A, f)$, a set of goal instances $G \subseteq P \times O$, and a belief marking $\mathcal{B}_n$.
output : Set of goal-oriented transitions T_g

1 $T_g \leftarrow \emptyset$
2 $O_m \leftarrow \emptyset$ // Set of relevant token types
3 $A_{\text{in}} \leftarrow \{(t,p) \in A \mid t \in T_c\}$ // Incoming arcs connected to the agent
4 $A_{\text{out}} \leftarrow \{(p,t) \in A \mid t \in T_c\}$ // Outgoing arcs connected to the agent
5 **foreach** $(p,t) \in A_{out}$ **do**
6 $\quad I_G \leftarrow \text{out}(t) \cap G$ // Goal instances generated by t
7 $\quad$ **if** $I_G \neq \emptyset$ *and* $\mathit{active}_{\mathcal{B}_n}(t) > 0.5$ **then**
8 $\quad\quad T_g \leftarrow T_g \cup \{t\}$
9 $\quad\quad O_m \leftarrow O_m \cup \{o \mid (p', o) \in I_G\}$
10 **foreach** $(t,p) \in A_{in}$ **do**
11 $\quad I_G \leftarrow \text{in}(t) \cap G$
12 $\quad$ **if** $I_G = \emptyset$ *and* $\exists o \in O_m, p' \in P\colon (p', o) \in \mathit{out}(t)$ *and* $\mathit{active}_{\mathcal{B}_n}(t) > 0.5$ **then**
13 $\quad\quad T_g \leftarrow T_g \cup \{t\}$
14 **return** T_g

Algorithm 2 requires the activeness of a transition $t \in T$ in $\text{active}_{\mathcal{B}_n}(t)$, defined by the probability mass of all markings that enable t:

$$\text{active}_{\mathcal{B}_n}(t) = \sum_{M \in \mathcal{M}'} \mathcal{B}_n(M) \text{ where } \mathcal{M}' = \{M \in \mathcal{M} \mid t \text{ is enabled w.r.t. } M\} \tag{5.13}$$

Algorithm 2 first iterates over outgoing transitions, i.e. place actions, of the agent in line 5 to find out which not yet fulfilled goal instances the agent can produce. Produced token types are collected in O_m in line 9. In line 10, Algorithm 2 iterates over incoming transitions but is restricted to relevant token types and excludes transitions that revert goal instances. The output of the algorithm is a subset of transitions that all contribute to the achievement of the task goal, no matter which one is selected.

Algorithm 2 is used in both the simulation experiments in Section 5.7 and the study in Section 7.3. Agent strategies to select one of the transitions from T_g can then differ. If multiple controllable agents are involved, the subset of T_c associated with the current agent is used in lines 3 and 4.

This concludes the description of the formal task model. Next, we sketch how the components are integrated into the software framework and run an evaluation to test the latency and accuracy of the approach.

5.6. Implementation

The software combines a publish-subscriber architecture design principle with the entity-actor framework ENACT [126]. We use the term actor here in a more general interpretation, where every program thread is an actor. In terms of the publish-subscriber architecture, each actor can be both a publisher and a subscriber. A publisher emits signals like the arrival of a new point cloud, the detection of an object, or the update of the task state. Subscribers are registered to listen to these events.

The algorithms in this chapter are assigned to three distinct actors, each with a specific role. The first actor receives and preprocesses the data from the camera to meet the assumptions from Section 5.2. It then emits signals of processed point clouds. The second actor takes the latest signal—ignoring intermediate signals if it cannot catch up with the processing speed of the first one—and runs the detection algorithm from Section 5.2. Detection results are stored in a shared, concurrently accessible storage. The storage guarantees that each actor uses the most recent information about an object. The third actor receives signals about completed actions of controllable agents. It periodically generates the emissions and runs the update procedure described in Section 5.4.

The evaluation code for the algorithms is also implemented for additional actors that receive signals from the third actor and access the shared storage. That way, we can achieve higher throughput compared to running all components sequentially in a single thread.

5.7. Simulation Experiments

We evaluate our algorithm (NET) on pick-and-place tasks of increasing complexity. The performance is compared with an implementation following the ageing-based approach (DECAY) described in Baraglia et al. [53].

5.7.1. Setup

The experiments are run on a desktop computer with $16\,\mathrm{GB}$ of DDR4 memory and an Intel i5-8600K CPU with six cores and a maximum speed of $4.1\,\mathrm{GHz}$. To conduct the experiments, a simulation environment has been implemented where two or more agents can pick and place small, monochrome toy blocks. The agents chose actions at random from their set of goal-oriented transitions (Section 5.5). A top-mounted, simulated depth camera observes the scene and generates a coloured point cloud. This point cloud is the input one would obtain from a real depth camera. Afterwards, we forward the point cloud to the processing pipeline described in Section 5.6. Based on the new marking, we conclude which objects are picked and placed. Since we can end up with multiple markings, it may not be apparent whether a particular pick or place action has occurred. Let M' be the set of markings that agree that $t \in T$ fired. We then *detect* t if the summed probability over M' is above $60\,\%$. A percentage above $50\,\%$ was chosen to avoid two mutually exclusive transitions being treated as fired.

The DECAY algorithm obtains the detected and classified objects of each frame as input. In line with the implementation in Baraglia et al. [53], occlusion information is not provided. Occluded objects are treated as not detected. For non-detected objects, the presence likelihood is decreased; for detected ones, it is increased to a limit of one. If the presence likelihood drops below 0.4, the object is removed. The decay rate is set to 0.1 per second. This means that an object with maximum presence likelihood is removed after it is constantly not detected for $6\,\mathrm{s}$. We associate the removal of an object with a pick action and the presence of a new object with a place action.

The simulation environment implements four benchmark tasks depicted in Fig. 5.3. Task B-1 consists of placing four coloured cubes in the centre of the table. Two agents execute the task in parallel. The task approximately represents the one used by Baraglia et al. [53] as task A1. Task B-2 consists of more objects, and the white cubes need to be stacked onto the red blocks. This results in more occlusion, and several objects of the same type exist in the scene. Finally, B-3 and B-4 are the most complex ones. In B-3 four agents work on the task and objects need to be swapped. In B-4 the same structure as in the human-robot study (Section 7.3) is constructed. Both benchmarks are characterised by longer-lasting occlusion. In all benchmarks, agents may place an object at a temporary location if they can no longer produce a goal instance, e.g. because another agent completed that action.

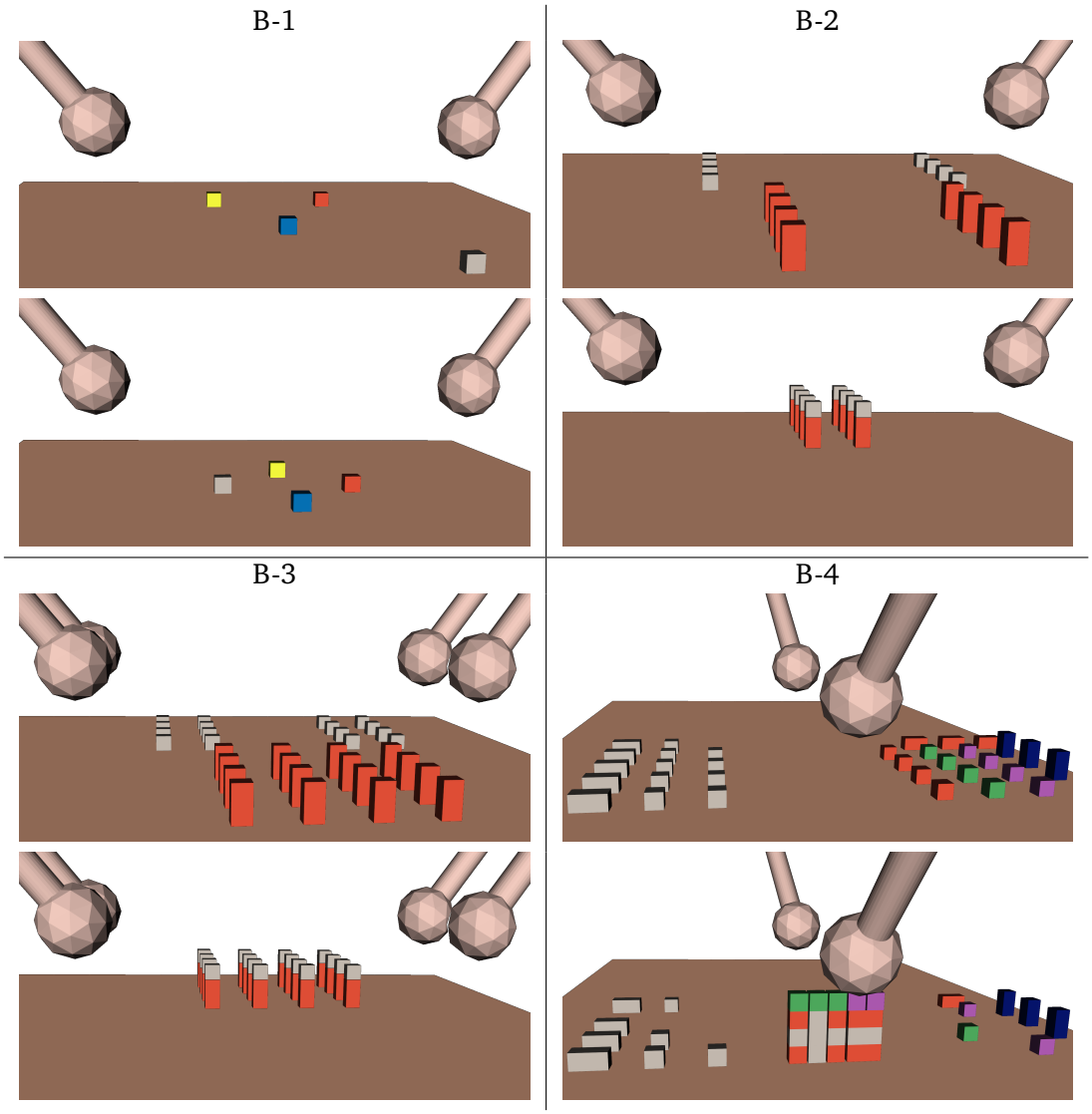
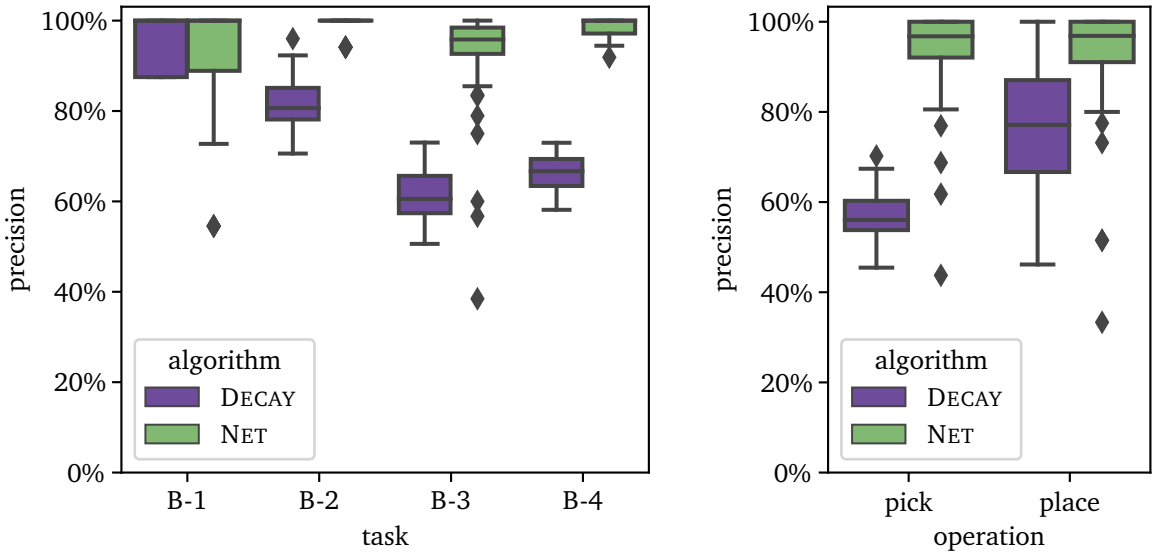


Figure 5.3.: Initial (top row) and final states (bottom row) for each of the four benchmark tasks. The skin-coloured balls represent the agent's hand. Each of them can pick, transfer and place blocks independently. When transferring blocks, the agents do it in a straight line, passing through other agents if necessary.



(a) Precision of DECAY and NET across tasks.

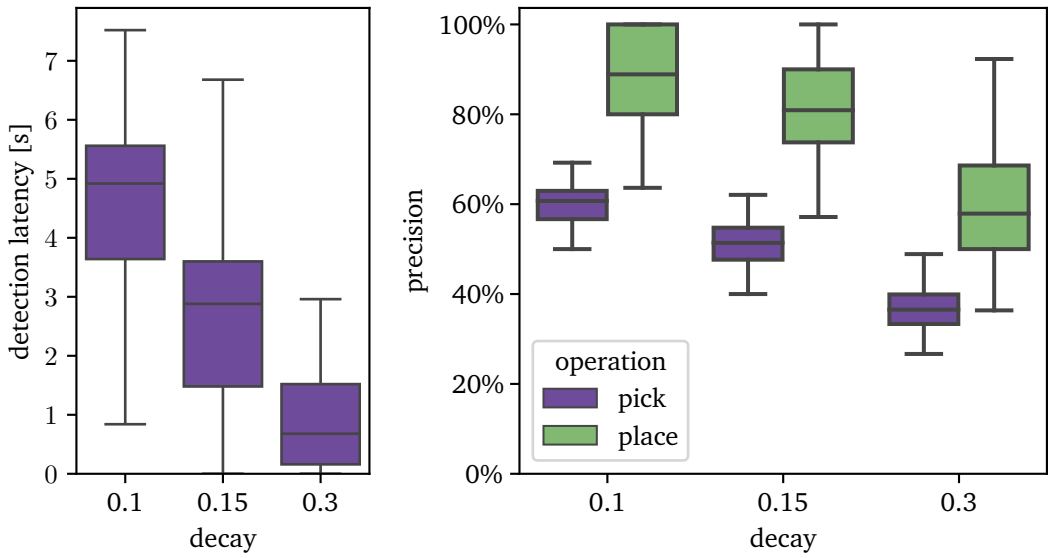
(b) Comparison of DECAY and NET on B-3 split by operation type.

Figure 5.4.: The algorithms are run multiple times. The precision is the number of correctly detected operations divided by the total number of detected and missed operations. Higher values indicate better performance.

5.7.2. Results and Discussion

Each task was executed 50 times. After each task is completed, the state of the simulated world and algorithms are reset. In each execution run, the detected actions by both algorithms are recorded and compared with the ground truth. The algorithms can (i) correctly detect an action, (ii) detect an action that was not executed, or (iii) miss an executed action. The latter two occasions count as errors. For each run, we calculate the *precision* of each algorithm by dividing correct actions by total actions (i.e. correct ones plus errors). In case of a false positive, the algorithms often correct their model by detecting an action that reverses the incorrect detection. These corrective detections are ignored when calculating the precision.

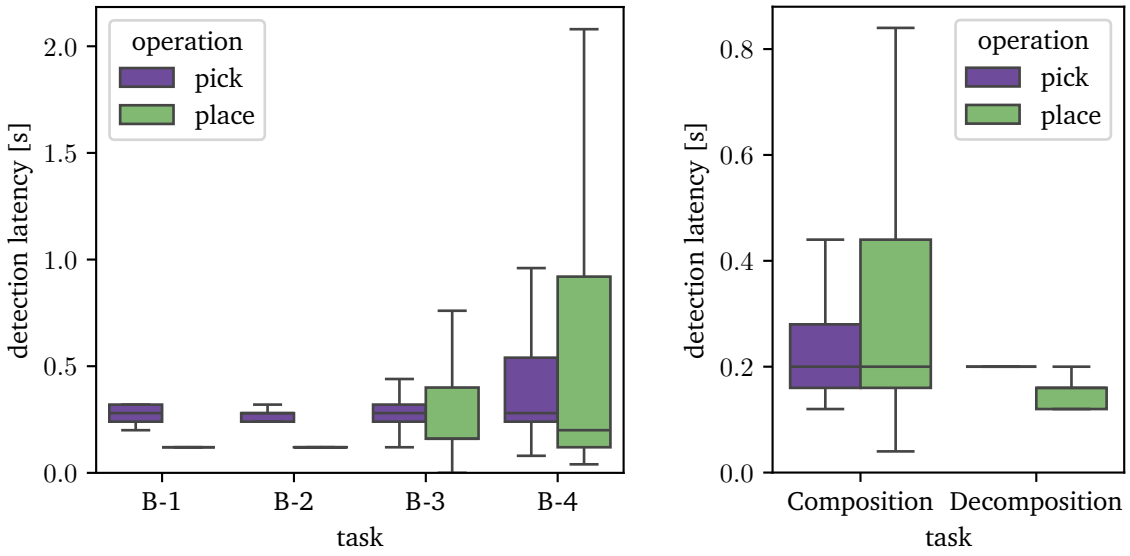
Figure 5.4 compares the algorithms against each other. NET achieves an average precision above $95\,\%$ on all tasks. In contrast, the average precision of DECAY ranges from $60\,\%$ to $97\,\%$. In B-1 alone, both algorithms perform roughly equally. In all other tasks, the average precision differs by at least 20 p.p. in favour of NET, with the largest difference of 34 p.p. for B-3.



(a) DL (only pick actions). Lower values are better.

(b) Precision of DECAY on B-4. Higher values are better.

Figure 5.5.: Detection latency and precision for varied decay rates of DECAY on B-4. The decay rate for the presence likelihood is varied and plotted along the x-axis. The decay rates of 0.1, 0.15, and 0.3 correspond to an expected value of 6, 4, and 2 seconds to detect the removal of an object. Lower latency values are possible because the decay starts with occlusion, which occurs before the operation completes.



(a) Comparison across tasks and operation types.

(b) DL on B-4. The structure is repeatedly composed and decomposed without resetting the state of the NET algorithm.

Figure 5.6.: Detection latency of NET algorithm in seconds. Lower values are better.

The difference is particularly noticeable in situations with high occlusion. The explicit handling of occlusion prevents the algorithm from incorrectly assuming that the object is gone. It sometimes struggles in the B-1 benchmark if the colour of the object is not correctly detected. Then, incorrect place actions are detected. Since B-1 involves only a few objects, a single error decreases precision by $12.5\,\%$.

A comparison of pick and place actions in B-3 (Figure 5.4b) reveals that NET performs identically well for both types. The DECAY algorithm produces more errors for pick actions than for place actions. The main source of errors is occlusions that incorrectly trigger the detection of a pick action. Missed actions only play a minor role of less than $2.5\,\%$.

Finally, we compare the speed of both algorithms. We do this by calculating the *detection latency* (DL) of each operation. The DL measures the time interval between action completion and algorithm detection. An action is considered completed when the simulated agent achieves a hover pose above either: (i) a picked object, or (ii) a placement location.

The DL for the DECAY algorithm remains the same over all tasks. For place actions, DL remains below $100\,\mathrm{ms}$—the duration until the hand uncovers the object. The DL of

pick actions is determined by the decay rate. As depicted in Figure 5.5, increasing the decay rate from 0.1 to 0.3 reduces the DL of pick actions from $5\,\mathrm{s}$ on average to below $1\,\mathrm{s}$. Place actions are not affected and always detected within $0.1\,\mathrm{s}$. However, the increase in decay rate significantly reduces the detection precision by 20 p.p. of both pick and place actions.

The NET algorithm can detect pick actions within $300\,\mathrm{ms}$ and place actions within $0.2\,\mathrm{s}$ for the simple tasks. Figure 5.6a shows a comparison across tasks. When complexity and occlusion increase, however, the variance of DL increases, too. Then, more transitions need to be tested. Even outliers of several seconds could be observed. This happens when the algorithm cannot find a valid marking. A frequent reason is that some places are incorrectly detected as present, e.g. because the hand is on the same height as the block. Still, the longer reasoning effort has the benefit of better consistency of the task state and thus better precision, as shown in Figure 5.4.

A closer inspection of DL in B-4 (Figure 5.6b) yields that NET detects most actions within $250\,\mathrm{ms}$, rarely exceeding $400\,\mathrm{ms}$. For the decomposition, place actions take longer to be detected, and for decomposition, pick actions. This clearly shows the effect of occlusion. In contrast to B-3, all place locations are within a small space.

5.8. Conclusions and Discussion

This chapter makes two conceptual contributions towards robust task state tracking. The first one is a formal model derived from coloured Petri nets to model task flows for dynamic human-robot teaming (Section 5.3). The model encodes the current state and position of each workpiece as well as object allocation to agents. It uses places to encode physical locations of the workspace and hands/grippers, coloured tokens for different types of objects, and transitions for actions. Each place can contain at most one token, and arcs have exactly one colour. Moreover, a special empty token exists. This is used for unstacking transitions such that only the top-most object can be picked. Transitions are divided into controllable (executed by the robot) and uncontrollable (executed by the human). The model is capable of expressing parallel execution, alternatives, reversal, and loops. Since reversal of actions and alternatives pose challenges to identifying the next executable action, we present a novel and lightweight filtering algorithm (Section 5.5). The filtering algorithm returns all transitions that are both executable and contribute towards the goal.

An important design aspect of the model is that uncontrollable transitions cannot be observed directly, but only their effects on places. Fired transitions and markings must

therefore be constructed from what is observed by the camera. Tracking the task state is therefore the second conceptual contribution of this chapter. The tracking addresses the challenges of occlusion and measurement noise arising from real-world scenarios. A recognition pipeline is set up. The camera image is processed to detect and classify known objects using hand-crafted features (Section 5.2). The object detection outputs occlusion and occupancy information of predefined locations, which are input to the task state tracking.

We present an algorithm for state tracking (Section 5.4.2) consisting of three steps. The first step updates the belief marking in terms of completed actions from controllable agents. The second step filters the transitions that could have fired given the previous and current emissions. The final core step utilises a Monte-Carlo-based procedure to sample sequences of fired transitions. The resulting markings are compared with the current emission. If they match, the transition sequence and marking are added to the candidate set. In the end, the number of samples per marking determines a probability distribution. This probability distribution represents the task state known to the robot and can be used to calculate the goal-oriented actions for the robot. The sequence of fired transitions is recorded to derive insights about human preferences in Chapter 6.

If observations are contradicting, no updates are made until a valid marking can be recovered. This conservative approach minimises the incorrect detection rate below $5\,\%$ on average. Still, the overall pipeline has a low latency of $250\,\mathrm{ms}$ on average. The explicit detection of emptiness makes it faster than ageing-based approaches that require an object to fade out for several seconds.

Despite its flexible design, the presented approach has a few drawbacks: (i) Object detection relies on hand-crafted features. This requires manual parameter tweaking to find the optimal balance between false positives and false negatives concerning the presence of objects. The issue with false positives is that the algorithm temporarily produces an invalid marking until the incorrect detection has vanished. Negative effects of the invalid marking are the detection of invalid actions (as mentioned in Chapter 6) and the communication of the task state to the user (Section 7.3). Object shapes are limited to primitive toy blocks. The input to the task state tracking was therefore designed to be a generic interface. Still, it remains subject to further investigation how well the tracking works if the input originates from deep neural networks that can be trained to detect the state of custom assemblies, see e.g. Chen et al. [116] or Pelosi et al. [114].

(ii) The design of the task model is targeted towards pick-and-place assemblies where the target state is largely defined by the positions of workpieces. Other actions such as placing and fixing screws, applying sealing, etc. can be encoded in the model but

would require the introduction of new token types to model intermediate states and perception components to detect these token types. Besides the action type, two more constraints are inherent to the model. The first one is the limitation of at most one token per place. Typical assembly workstations use for instance boxes where the worker picks screws. These boxes can currently not be modelled as a single place with many tokens. The second constraint is that the positions of places are fixed. This holds true for firmly clamped workpieces, but is no longer applicable if the workpiece can be rotated or moved. Future research could therefore review common action types in industrial assembly scenarios and their execution constraints to extend the proposed model accordingly.

(iii) Task models are specified and implemented in source code and require extensive expert knowledge. Future extension points are the automated generation from common task modelling formalisms, such as assembly sequences derived from CAD models [127] or user-designed precedence graphs [17].

CHAPTER 6

Learning Human Preferences

A core challenge in fluent human-robot teaming is the adaptation to the strategy the human wants to execute a task. Robots should therefore be capable of quickly and reliably learn the behaviour rules specific to a teaming partner [37]. The previous chapter developed a method to robustly track the task state and recognise executed human actions from it. This section builds up on the recognised action sequences:

Domain	Datasets
Body gestures	**KTH** [128], **UT-Interaction** [129]
Daily household routines	**Breakfast** [130], **50Salads** [131], **Setting the table** [132], **IKEA-ASM** [133], **CAD-120** [134]
Instruction videos	**COIN** [135], [136]
Business Process Management	**BPIC12** [137], **Helpdesk** [138]
Industrial Domain	**AssemblyHands** [139] (toy assembly), **OpenPack** [140] (packaging)

Table 6.1.: Frequently used datasets for action prediction grouped by application domains.

It proposes a model to predict future human actions. To be precise, we want to get a probability distribution over the next action given the previous actions. This brings the robot closer to forming the shared mental model described in Chapter 3. Predicting the human's actions allows the robot to be more helpful, supportive, and to avoid disturbing actions.

Before diving into this topic, we differentiate action prediction from related terms and clarify the problem. Related to action prediction are action recognition, intention prediction, and intent prediction:

Action Recognition : 'recognize a human action from a video [or other form of input information] containing complete action execution.' [141]

Action Prediction : 'reason a human action from temporally incomplete video data [or other form of input information].' [141]

Intent Prediction : 'simultaneously inferring the action/interaction class and generating the involved persons' future body motions.' [142]

Intention Prediction : 'extend an incomplete sequence of actions to its most likely intended goal.' [143]

The key distinction between recognition and prediction is that part of the information constituting an action has not yet been observed. Chapter 5 only detects the completion of an action. Information about started actions is thus not available. Both action and intent prediction focus on short-term predictions. Intent prediction, however, emphasises the motion trajectory as the main outcome (e.g. [144, 145, 146, 147, 148]). The type of next action has a subordinate role or is neglected altogether. Longer-term predictions, constituting several actions or referring to some goal or state, are referred to as intention prediction. In the context of assembly, this can be the distinction of which task the user started (e.g. [149, 150]) or the cooperation state of the human (e.g. cooperating, pausing, or doing another task [151]).

The industrial application domain, however, poses two additional requirements on the model. First, a long training session to adapt to the human teaming partner would diminish productivity gains. Training should therefore be achieved with little training data (referred to as *few-shot learning* [152]). Optimally, the training is conducted while the system is running—to adapt to the worker while interacting with him or her (referred to as *incremental learning* [153]). Second, large datasets of related assembly tasks are not available either (Table 6.1). For other application domains, such as household kitchen

[154], autonomous driving [155], daily actions [141], and business processes [156] datasets have been available for some years. However, these datasets are designed to target action recognition in general rather than action prediction for specific human subjects. The domain of action prediction for industrial tasks is relatively new, with larger datasets emerging only in the past two years to support research in this area. However, none of these existing datasets align with our specific requirements: high flexibility in task execution and workpieces fixed to the workspace, e.g. not hand-held (Chapter 5). Confidentiality and privacy are two of the reasons why the availability of data on industrial assembly processes is limited (Table 6.1). As such, incremental few-shot learning without related datasets or models for knowledge transfer is the frame condition for this chapter.

The chapter is structured as follows. First, we review related work in terms of problem structures and applied methods (Section 6.1). We included related application domains to incorporate insights that can be applied to our domain. Section 6.2 provides a formalisation of general assembly rules. This serves as a fallback for our later algorithm and as a baseline for the evaluation. Next, an action embedding is constructed to have a numerical representation of actions that still encodes important semantics (Section 6.3). The embedding takes inspiration from the general assembly rules. Finally, the structure (Section 6.4) and the training process (Section 6.5) of the prediction model is described. How to make long-term predictions is explained in Section 7.2 in conjunction with the robot's decision-making. The evaluation is then presented in Section 7.5.

6.1. Related Work

Problem structures can further be categorised in terms of input, output, and available information on the task structure. The most prominent input domains are video streams (sometimes extended with depth information) [141], motion trajectories in terms of sequences of keypoints [157], and discrete event logs [156]. Video streams contain a high level of information with nuances about how the task is executed. However, this comes at the cost of low information entropy per information unit (pixel). Video streams are thus considered unstructured information as they are not directly interpretable by computers. Event logs are the exact opposite. They represent structured, condensed information that is easy to preprocess. Depending on the use case, event logs are either restricted to an id, timestamp, and action label per entry—or they can be augmented with additional information. Motion trajectories play an intermediate role. Their information is more condensed than videos but still has continuous fine-grained motion, in contrast

to discrete event logs. To obtain motion trajectories, either the video stream must be preprocessed or markers must be attached to the tracked object. Since video streams require large pre-trained networks for feature extraction, approaches in the human-robot collaboration (HRC) domain have restricted the input to motion trajectories (e.g. [145, 146, 148, 158, 159]). Preliminary approaches using video input with an industrial setting exist (e.g. [160, 144]) but are far from integration into HRC settings. Still, the aforementioned approaches require several hours of labelled training data[1].

The two most prominent target variables are the next action label (e.g. [160, 158, 161, 162, 163]) and the timing of the next action (e.g. [164, 107, 41, 165, 163, 166, 167]). For both target variables, the exploitation of information about the task structure plays an important role. In the presence of execution constraints, action prediction can either take the form of learning these constraints or learning preferences. We enforce a strict separation between those two concerns and assume the first one to be given or computable (see goal-oriented actions in Section 5.5) and only want to learn the second one. In previous research in the HRC domain (e.g. in [168, 169]), these two aspects have been convoluted. Models were trained on simulation runs of the task and evaluated against real-world demonstrations. Accuracy was then calculated against all possible actions. This demonstrates that the model can learn the already known task constraints, but provides no insights into how human preferences are captured.

The most prominent **methods** are *recurrent networks* (RNN) in event prediction [156], with *Long-Short Term Memory Networks* (LSTM) being the most widely applied subgroup. Other prominent methods include *Convolution Neural Networks* (CNN), stochastic models, and variants of *Markovian Decision Processes* (MDP). Most neural networks consist of three to five layers. The last layer is a fully-connected layer that forms a classifier outputting a probability distribution over all actions. All layers are trained from scratch. To cope with input sequences of varying length, most approaches utilise LSTM layers (e.g. [167, 170, 163, 160]). Fully-connected and convolutional neural networks are used as well (e.g. [171, 172, 173, 174]). In terms of prediction accuracy for event logs, Neu, Lahann and Fettke [156] have pointed out that feed-forward neural networks can achieve the same accuracy as LSTM networks. LSTM networks might offer the benefit of variable input lengths, but suffer from vanishing gradients if more than one LSTM layer is used.

In the application domain of human-robot cooperation, stochastic models (e.g. [107, 164]) or variants of Support Vector Machines (e.g. [159]) are used to learn human

[1] Amount of training data: 200 human-human training sessions [158], 15 subjects executing the task in four conditions [145], 3 h of demonstrations [146], 3 h of ten subjects [148, 132].

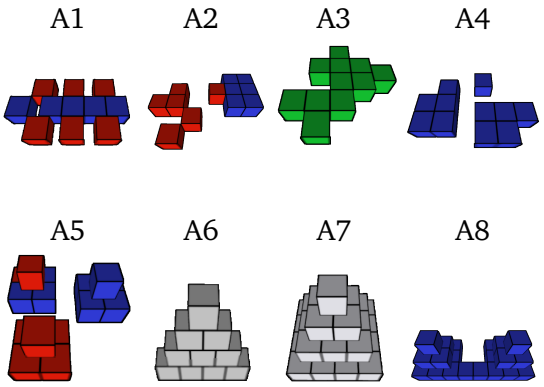
preferences. However, input to their algorithms is either the full state of the workspace (e.g. [164]) or the full sequence of previous actions (e.g. [107, 159]). This limits the applicability of these approaches to assembly tasks with a very small set of human actions and states. Otherwise, the required amount of training data would grow too much.

A special case is mixed approaches of action or intention prediction combined with robot decision-making. These approaches exist on two abstraction levels. In the low-level case, continuous trajectories are processed. In the high-level case, discrete actions are processed. Input to low-level approaches is the human hand trajectory, and they output the robot's motion. The approaches are trained offline using inverse reinforcement learning on datasets of executions with different strategies. That way, preferences are implicitly learnt from the trajectories and used to derive the robot's motion. Examples are box painting where the user paints a box held by the robot [145], object disposal where one agent opens a drawer and the other one places an object in it [175], box packing [158], and toy car assembly where the robot holds the chassis in a pose to assist assembly [176]. Major drawbacks of these approaches are the acquisition of sufficient training data[2], playing the robot's part during training, and ensuring the stability of the reinforcement learner.

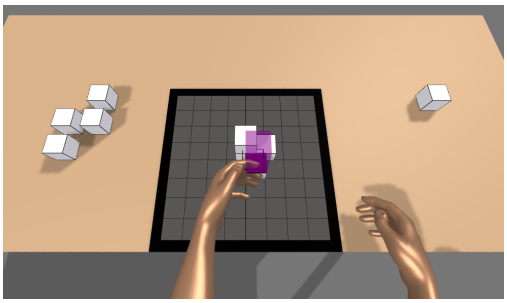
In the high-level case, *Partially Observable Markovian Decision Processes* (POMDPs) are used. Besides other components, POMDPs define a set of states of the environment (*state space*), a set of observations, and a set of actions for a controllable agent, i.e. the robot. In the POMDP formalisation, the robot does not know the exact state of the environment but only receives partial information about it in terms of observations. The unobservable state space encodes the known world state combined with the unknown human's intentions. Human actions are encoded in the observations, as these cannot be directly controlled. POMDPs suffer from large state and action spaces that require many parameters to specify the transition probabilities. Approaches therefore use hand-crafted, pre-initialised probability distributions (e.g. [177, 48, 178]).

From the literature review, we conclude that action prediction for human-robot teaming is still in an early stage. From the found approaches, most utilise POMDPs, reinforcement learning, or stochastic models. Reinforcement learning is mainly deployed when human and robot motion trajectories must be learnt and optimised in conjunction. Their prediction horizon is rather limited, and they require annotated and segmented training data. POMDPs are often initialised with hand-crafted parameters due to their complexity. Both POMDPs and stochastic models do not scale well if tasks become a bit more complex.

[2]Amount of training data: 200 human-human training sessions [158], 15 subjects executing the task in four conditions [145].



(a) Tasks A6 and A7 are identical to the ones in Mayer et al. [180].



(b) The participant's perspective on executing the task. Additionally, one of the task images from the left figure was displayed to indicate the final state of the task.

Figure 6.1.: Tasks and setup of the online study [182].

Recurrent neural networks, often deployed in event processing, suffer from the same problem that they require large training data for complex tasks. Multi-layered neural networks fit the requirement of incremental few-shot learning since they have proven to be successful, can cope with contradicting training data [179] and can be tuned in complexity to reduce training effort.

6.2. Encoding of General Assembly Preferences

The way humans execute an assembly task has been studied in the literature: Mayer et al. [180] let participants construct pyramids from Lego bricks and observed their brick placement strategies. They postulated and verified the following rules:

R1 The first assembly brick is placed in the left corner of the subject's field of view.

R2 Bricks that can be placed in the direct vicinity of bricks already positioned during a certain assembly step are preferred.

R3 The target object is assembled in layers parallel to the mounting surface.

R4 If the assembly consists of sub-assemblies of different colours, then these are completed in favour of layer-wise construction [181]. The rule has been confirmed in a different context where participants had to arrange objects of different colours [148].

A student's work [182] conducted an online study to verify and expand on the above findings. The study was conducted via Amazon Mechanical Turk in a microworld depicted in Figure 6.1b. Participants got a tutorial on how to control the microworld. They could change the view and use the mouse to control the hands, pick, and place blocks. After the tutorial, the participants executed ten tasks depicted in Figure 6.1a. The first row of tasks resembles a single layer and focuses on non-symmetric structures. Tasks A5 and A8 have multiple subcomponents consisting of more than one layer. Tasks A1, A2, and A5 offer blocks in different colours, but it is up to the participants how to arrange the colours in the final assembly. For tasks A6 and A7, there were two variants where users could control only one or both hands of the avatar in the microworld.

The study had 34 participants with an average age of $28.5\,\mathrm{years}$. All participants came from Europe and had an educational background ranging from being a student to having a master's degree. Five participants were removed because they quit the tutorial early,[3] and three because they needed too much time.[4] After removing those participants, 26 remained. User behaviour was then compared with the rules above. A first finding is that in the case of less symmetric task structures, less than half the participants started with the top left one as their first block (R1). The rules of spatial coherence (R2) and layer-wise construction (R3) were, however, strictly adhered to. Colour similarity (R4) played a negligible role. In case of spatially distinct subcomponents, participants preferred to complete the subcomponent before moving to the first layer of the next structure. An investigation into how the lower platform of A7 was constructed shows that the row-by-row strategy is dominant in the absence of specific colours or patterns. A smaller group of participants followed a snake-like strategy where the outer seam is completed first.

Susanto, Purwaningsih and Kurniawati [183] show that R2 and R3 are in line with humans' mental models. The authors presented half of the assembly sequence of a pulley to a group of Indonesian people. They let them predict which assembly step is executed next. Compared with other assembly sequences derived from real human observations, the one following the two rules achieved the highest prediction accuracy. We therefore conclude that R2 and R3 are the most dominant rules. R1 might not be that prevalent for all assemblies, but it is still the seed for the left-to-right row-by-row strategy. We therefore encode all three of these rules and give them equal weight. We still incorporate R4 if the first three rules are not discriminative enough, but give far lower weight to it.

[3] The study author assumes that this indicates repeated participation.

[4] They took more than one standard deviation longer than the others, indicating problems with task execution or longer pauses in between.

Algorithm 3: Utility functions for Algorithm 4

Function `projectedLength`(v, v_{br})**:** // Length of v after projection onto v_{br}
  if $\|v\| < \varepsilon$ **then**
    return 0
  $v_d = \langle v_{br}, v/\|v\| \rangle$
  if $v_d \leq 0$ **then**
    return ∞
  return $\|v\| / v_d$

Function `weight`(d_{min}, d)**:**
  return $0.5 \cdot \text{bell}\left(\max\left\{0, d - d_{min}\right\}, d_{min}\right) + 0.5$

Let A denote the set of all actions. We encode the rules in terms of a deterministic algorithm that, given the previous action $a_0 \in A$ and a set $A' \subset A$ of candidates for the next action, outputs a non-normalised distribution P over A'. An action is characterised by a workspace location, size vector and luminance value (all derived from the manipulated object). We assume a global coordinate system where the z-axis is in the opposite direction of gravity. Let $f \in \mathbb{R}^3$ and perpendicular to the z-axis denote the viewing direction of the human. This is necessary to derive the notion of left and right. The position $p_a \in \mathbb{R}^3$ of a pick or place action a is given by the centre location where the object is picked from or placed. The top-leftmost action a_{tl} of a set of actions A is thus given by $a_{tl} = \arg\min_{a \in A} \langle R_z(45°) \cdot f, p_a \rangle$, where $R_z(\cdot)$ is a rotation around the z-axis. We then model a two-dimensional Gaussian distribution around the top-left object. The standard deviation of the distribution is defined by the object farthest away: $\sigma = 0.5 \cdot \max_{a \in A} \|p_{a_{tl}} - p_a\|$. The probability that a participant starts with an action $a_s \in A$ is thus modelled by:

$$\mathcal{P}(a_s) \propto \text{bell}\left(\|p_{a_{tl}} - p_{a_s}\|, \sigma\right) \cdot 2^{p_a[2] - z_{min}} \tag{6.1}$$

where $z_{min} = \min_{a \in A} p_a[2]$. Here, $p_a[2]$ denotes the z-component of p_a. The exponential term on the right lowers the probability of objects in higher layers to account for R3. If previous actions are available, we run Algorithm 4 twice with the previous two actions and sum the probabilities. Normally, the second-to-last is of the same type and given normal weight in Algorithm 4. However, in some circumstances—e.g. if there is only one previous action or the subcomponent is completed—we can only derive some information

Algorithm 4: General Prediction Rules

input : A set $A' \subset A$ of candidates for the next action, where A is the set of all actions, the previous action $a_0 \in A \setminus A'$, a function $p\colon A \to \mathbb{R}^3$ returning the location, a function $d\colon A \to \mathbb{R}^3$ for the size vector (i.e. diagonal fo the bounding box) and a function for the luminance $l\colon A \to [0,1]$. All three functions refer to properties of the object manipulated by the given action.

output : Non-normalised distribution $P\left(a \in A' \mid a_0\right)$

/* Setup-specific, normalised vector pointing from top left to bottom right of workspace. It spans a one-dimensional subspace (br-space) where top left actions have lower values than bottom right ones. */

$v_{br} \leftarrow R_z(0.3\pi) \cdot -f$

$P_{xy} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

// Smallest value of an action in br-space

$d_{br} \leftarrow \min_{a \in A'} \{\texttt{projectedLength}\left(P_{xy}p\left(a\right), v_{br}\right)\}$

// Distance of the action closest to a_0 in x-y-plane (excluding those with distance 0)

$d_0 \leftarrow \min_{a \in A', \|P_{xy}p(a) - P_{xy}p(a_0)\| > 0} \{\|P_{xy}p\left(a\right) - P_{xy}p\left(a_0\right)\|\}$

// Action closest to a_0 in br-space (excluding those with distance 0)

$a' \leftarrow \arg \min_{a \in A', \|P_{xy}p(a) - P_{xy}p(a_0)\| > 0} \{\|P_{xy}p\left(a\right) - P_{xy}p\left(a_0\right)\|\}$

$z_{min} \leftarrow \min_{a \in A'} \{p\left(a\right)[2]\}$ // Lowest action in z-direction

foreach $a \in A'$ **do**

  $w_1 \leftarrow \texttt{weight}\left(d_{br}, \texttt{projectedLength}\left(P_{xy}p\left(a\right) - P_{xy}p\left(a_0\right), v_{br}\right)\right)$ // R1

  $w_2 \leftarrow \texttt{weight}\left(\left\|P_{xy}a' - P_{xy}p\left(a_0\right)\right\|, \left\|P_{xy}p\left(a\right) - P_{xy}p\left(a_0\right)\right\|\right)$ // R2

  $w_3 \leftarrow \texttt{weight}\left(\frac{1}{2}d\left(a_0\right)[2], p\left(a\right) - \frac{1}{2}d(a)[2] - p\left(a_0\right) - \frac{1}{2}d\left(a_0\right)[2]\right)$ // R3

  // Low preference for R3 & volume, $\mathrm{vol}(\cdot)$ denotes the product of vector components

  $w_4 \leftarrow \frac{1}{2} + \frac{1}{4}\,\mathrm{bell}\left(l(a) - l\left(a_0\right), 0.02\right) + \frac{1}{4} \cdot \mathrm{bell}\left(\mathrm{vol}\left(d(a)\right) - \mathrm{vol}\left(d\left(a_0\right)\right), 0.02\right)$

  if *a and a_0 are the same action type* **then**

    $P'\left(a \mid a_0\right) = w_1 \cdot w_2 \cdot w_3 \cdot w_4$

  else

    $P'\left(a \mid a_0\right) = \frac{1}{4} \cdot w_1 \cdot w_2 \cdot w_3 \cdot w_4$

// subtract lowest action weight

$P\left(a \mid a_0\right) = P'\left(a \mid a_0\right) - \max\left\{0.01, \min_{a \in A'} P'\left(a \mid a_0\right)\right\} \;\forall a \in A$

return P

from the previous action. Therefore, we do the same calculation for different action types in Algorithm 4 but give it a lower overall weight.

This concludes the formalisation of the algorithm for $P(A')$ based on fixed assembly rules. We refer to the formalisation in Section 6.5. Section 6.3 explains how the input is prepared before our custom-designed neural network is introduced.

6.3. Feature Space Construction

A crucial step before applying any prediction model is the construction of a feature space that can encode all relevant information. Each action must be encoded into some numerical vector such that it can be input into neural networks. Frequently used encodings for categorical data are one-hot vectors and embeddings [156]. *One-hot vectors* have one dimension per category. An encoding of an action is represented by a 1 in the corresponding dimension and 0 otherwise. One-hot vectors are easy to construct but can suffer from the curse of dimensionality. In [156], the problem was not negligible for 20 categories[5].

Another drawback is that action similarities cannot be encoded. This problem is solved by embeddings. *Embeddings* construct a lower-dimensional feature space than one-hot encodings by placing related actions close to each other. However, these embeddings need to be trained alongside the predictor (e.g. [170]) or constructed from other datasets. If actions can be fully specified by words, established word representations from natural language processing can be utilised to construct the feature space (e.g. [184]). Nemlekar et al. [185] propose a hand-crafted feature space that encodes part switching, tool switching, physical and mental effort for each action. The core challenge for our task setup is, however, how to represent the state of the workspace in a condensed form. We pursue a two-fold approach: The embedding is hard-coded and incorporates additional information from the state of the workspace. However, we also train the neural network to perform dimensionality reduction on part of the feature vector.

The assembly rules R1 to R4 highlight which information is important for predicting assemblies, namely spatial information, colour, and size. Spatial information is the combination of position and neighbourhood information. A prerequisite to computing spatial information is that each action has a spatial anchor. That means, for each action, there is a particular object and position in the workspace.

[5] To encode all actions of the benchmark tasks described in Section 7.3 a feature space of dimension 134 is required

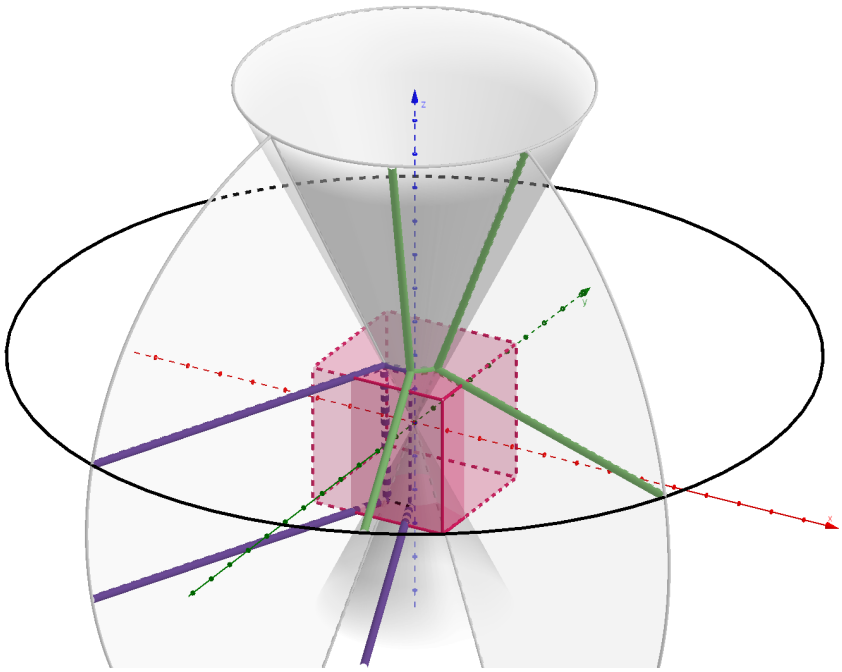


Figure 6.2.: Partitioning of the workspace around the bounding box of a manipulated object (magenta in the centre). The grey surfaces show some of the boundaries.

The main idea of this part of the work is to encode actions in terms of these features. The feature space has 33 dimensions, where the first 27 encode neighbourhood information. All values are within the interval $[-1, 1]$ to match the usual input of neural networks. This means every feature is encoded relative to some maximum value. We start by explaining the remaining six ones. Colour is encoded as a single continuous variable representing the mean grey value of the object manipulated in the sRGB colour space. Size refers to the volume of the object relative to the largest object. Position information is encoded in an absolute coordinate system rooted in the base of the robot. The scaling is relative to the maximum extension of the workspace. The last component of the feature vector is a binary indicator of the action type with -1 for place and $+1$ for pick actions. Its purpose is explained in Section 6.4.

Neighbourhood information is implemented in terms of a three-dimensional polar coordinates-based density map of objects in the workspace. First, the workspace is partitioned into 27 subregions with the manipulated object in the centre. Figure 6.2 shows the partitioning. The top and bottom planes of the bounding box divide the neighbourhood into three layers. The black circle shows the separation between the top and middle layer. The circle is divided into eight slices, where each defines the

Algorithm 5: Calculate neighbourhood

input : A set W of objects in the workspace, the manipulated object $w \in W$, and two functions $p, d\colon W \to \mathbb{R}^3$ returning an object's position and diagonal in meters.

output : Neighbourhood information $N = \left([0,1]^i\right)_{i=0,\dots,26}$

$N[i] \leftarrow 0 \quad \forall i \in 0, \dots, 26$

foreach $w' \in W \setminus \{w\}$ **do**

  $v \leftarrow p(w) - p(w')$

  $v' \leftarrow \exp\{-\|v\|^2 / \left(2 \cdot d_m^2\right)\}$ // proximity measure where d_m is the distance threshold set to $5\,\mathrm{cm}$.

  if $\|v\| \leq \|d(w)\|/3$ **then** // w and w' overlap

    $N[9] \leftarrow \max\{N[9], 1\}$

    continue

  $z = \arccos\left(v[2] / \|v\|\right)$

  if $z \leq 0.464$ **then** // w' directly above w, angle between z-axis and $p(w')$ less than $26.6°$

    $N[0] \leftarrow \max\{N[0], v'\}$

    continue

  if $z \geq \pi - 0.464$ **then** // w' directly below w

    $N[18] \leftarrow \max\{N[18], v'\}$

    continue

  $r \leftarrow 1$

  if $p(w')[2] > p(w)[2] + 0.5 \cdot d(w)[2]$ **then** // w' above w

    $r \leftarrow 0$

  if $p(w')[2] < p(w)[2] - 0.5 \cdot d(w)[2]$ **then** // w' below w

    $r \leftarrow 2$

  $\phi \leftarrow \mathrm{atan2}(v[1], v[0])$ // Angle between x and v in x-y-plane

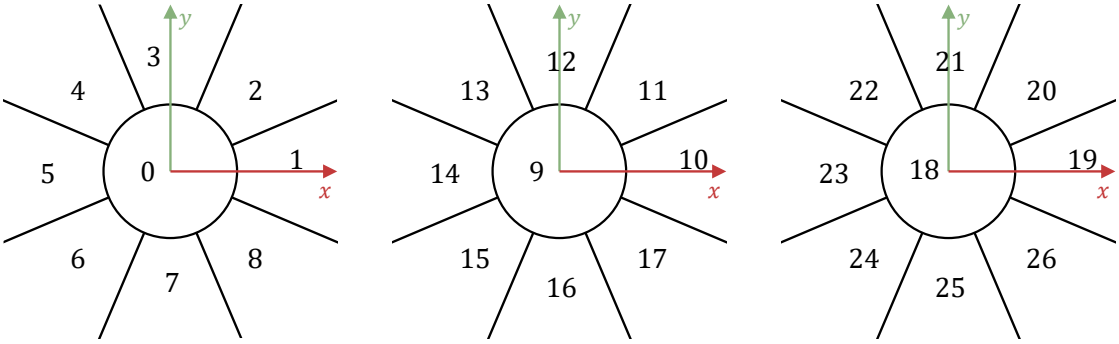
  if $\phi < -\pi/8$ **then**

    $\phi \leftarrow \phi + 2\pi$

  $i \leftarrow 9r + (1 + \mathrm{round}(4\phi/\pi))$ // Partitioning (Figure 6.3)

  $N[i] \leftarrow \max\{N[i], v'\}$

return N



(a) Top **(b)** Middle **(c)** Bottom

Figure 6.3.: View onto slices of neighbourhood partitioning of Figure 6.2. The slices are parallel to the x-y plane. The numbers represent the cell indices calculated in Algorithm 5.

sub-partitioning of the layers. The green lines show the boundaries of a cell in the top layer. The cell expands to infinity. The purple lines denote a cell in the middle layer. The space directly above and below the object is bounded by a cone.

All objects in the workspace belong to exactly one subregion. For each object, we use the centre of an axis-aligned bounding box to identify the subregion it belongs to. Algorithm 5 calculates the occupancy of each subregion. Line 4 calculates the density based on an exponentially decaying function. A distance of 0 yields a density of 1, and $5\,\mathrm{cm}$ gives a density of 0.5. Lines 5 to 20 calculate the layer of the subregion, and line 20 cf. the subregion index within a layer. The indexing of all subregions is depicted in Figure 6.3. If multiple objects are in a subregion, the maximum density value is used, i.e. only the closest object matters. This concludes the feature space construction.

6.4. Model Overview

The presented task model (Section 5.3) imposes restrictions on the actions to be executed. Not all actions can be executed in each state of the workspace. Thus, the number of candidate actions varies. One could simply define all possible actions as classes of the classification problem. But that would bloat the search space and leave the question open about how to handle inactive actions in the training data. If this issue is not addressed properly, the classification procedure merely learns task constraints as outlined in Section 6.1.

Taking the variable number of candidate actions into consideration, we formalise the prediction problem as follows. Let $n \in \mathbb{N}$ denote the number of previous actions we consider for prediction. Let $A \subset [-1, 1]^{33}$ denote the finite set of all actions and $A' \subset A$ the

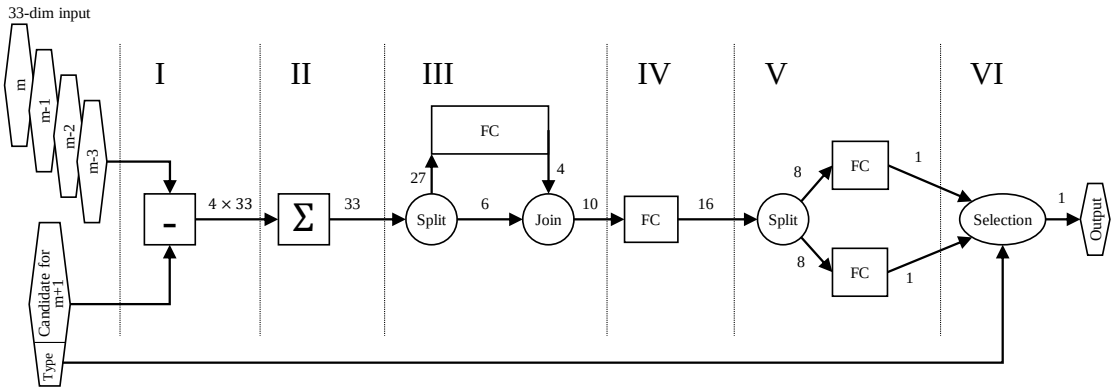


Figure 6.4.: Structure of the DNN. FC refers to fully connected layers with a sigmoid activation function. The numbers next to the arrows indicate the dimensionality of the data. The dashed vertical lines subdivide the stages.

subset of candidates for the next action. Here, A' corresponds to the set of active transitions given the current belief marking (Section 5.5) transformed into the feature space (Section 6.3). The problem is to predict a probability distribution $\mathcal{P}\left(a \in A' \mid a_1, \ldots, a_n\right)$ over A' given previous n actions $a_1, \ldots, a_n \in A$. We break down the problem into finding a weight function $w\colon A' \times A^n \to [-1, 1]$ with $w \propto \mathcal{P}\left(a \in A' \mid a_1, \ldots, a_n\right)$. Breaking down the problem has two advantages. First, input and output have fixed length, so that we can use a classical neural network. Second, training data can be constructed from action candidates only and inactive actions do not influence the latent space of the neural network. This section presents the architecture of the model, and Section 6.5 presents the training and execution.

The design of the *Deep Neural Network* (DNN) is inspired by expert systems, which define rules. A rule would consist of an if-condition derived from the previous action and an output characterising the next action. Since the task model splits the picking and placing of an object into two separate actions, we need to train the DNN for both action types. This can be achieved by training two separate networks for pick and place actions. However, interdependencies between pick and place actions can not be encoded, e.g. place close to the pick location.

I therefore opted for a combined approach: the DNN is split into a backbone part (stages I to IV in Figure 6.4) and two heads (stage V in Figure 6.4). The backbone extracts relevant features from the input data. The two heads encode the rules for pick and place prediction, respectively. The selection in stage VI ensures that the correct branch in stage V is used during evaluation and training. Inputs to the DNN are the feature vectors of the previous $n = 4$ actions (two pick and two place actions) and the

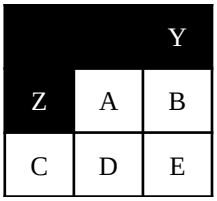


Figure 6.5.: Example for human placement pattern (viewed from the top). Black squares denote already placed blocks. White squares are slots where blocks need to be placed. Blocks Y and Z were placed in the previous steps.

candidate action. The action type of the candidate decides the selection in stage VI. The number four was chosen due to the bounded short-term memory of humans. Humans remember around four discrete items at a time as part of a complex task [186]. To guide the crafting process of the DNN, the prediction accuracy of simulated agents was visually inspected. Four deterministic agent behaviours were implemented to complete task B-2 in Figure 5.3:

(a) build left-to-right, placing white cubes immediately after the red blocks

(b) build left-to-right, but the red blocks first and then the cubes

(c) same as (a) but right-to-left

(d) same as (b) but right-to-left

The task was executed twice for each behaviour, and the DNN was trained online. Wire-frames indicated the output of the DNN to check how action probabilities are spatially distributed. For instance, the number of outputs in Stages III to V were determined that way. Their numbers were reduced until performance degraded, and then some margin was added to encode more complex tasks.

In the following, we describe the components of the DNN in detail. The following example is used to motivate and explain the design of each component: Assume the user builds a dense grid of 3×3 blocks, top to bottom, left to right—as depicted in Figure 6.5. The DNN basically has to encode the following two rules: (a) Place the block next to the previous one. (b) If the row is complete, place the next block to the left in the new row. To predict the next action, all place actions for empty slots of the grid are passed as candidates. In Figure 6.5, these are slots A to E. The DNN should then output the highest probability for A because it complies with the above rules[6]. But how can slot A be distinguished from B to E?

[6]We use the identifiers A to E to denote the slots, associated actions, and feature vectors depending on the context

In stage I, the candidate action is subtracted from the each previous action. Identical features result in zero, and therefore not influence the next step. This turns the position information into distances and the neighbourhood into density differences. The purpose of the subtraction is to discard uninformative information. In the example, the subtraction ensures that the next action in each row always has the same position offset of one block to the right and zero in both of the other directions. Moreover, when subtracting A from Z, we get a negative value for the left neighbour—making A distinct from B to E, which have a zero there.

Stage II is a weighted sum over feature vectors. The weighted sum allows the DNN to learn which of the previous actions is necessary for prediction. Weights are learned during training and normalised after each training run. Referring to the example in Figure 6.5, all weight would be given to the second previous action, which is the placement of Z. Thus, stages I and II are very simplified versions of what attention mechanisms [187] or Long Short-Term Memory units [188] do. However, using these advanced techniques would require many times more parameters than our entire DNN possesses.

Stage III performs the dimensionality reduction of the neighbourhood information. The 27 subregions are reduced to four features by a fully connected layer. The weight matrix is initialised in terms of Gaussian distributions with a different mean location in each row. This fosters the aggregation of neighbouring neighbourhood information and reduces the risk that two output features yield the same aggregation. The remaining six features skip this stage.

The remainder of the DNN is designed to encode conjunctions and disjunctions of predicates. These predicates are learnt in stages IV and V. Two layers with sigmoid activation functions allow to learn intervals. Referring to the example, stage IV can learn two features for the offset in the x-direction: at least one block to the right and at most one block to the right. In stage V, these are combined to the feature exactly one block to the right and given a high weight. The neighbourhood information can be used to give candidates with a non-negative difference for their top neighbours a higher weight. This gives slot A a better rating than D. The rule to start from left can also be encoded by giving an x-offset of two blocks to the left a high weight. That way, the rule only applies when being at the end (e.g. after placing Y). Due to the neighbourhood feature, slot Z gets a higher rating than slot C.

The final combined rating is output in stage VI. The output approximates the weighting function w for the given inputs $a \in A'$ (next action candidate), $a_1, \ldots, a_4 \in A$ (previous actions). Next, we show how to post-process the output of the DNN and how to generate training data.

6.5. Learning Procedure and Execution

The output of task state tracking described in Section 5.4.2 provides the executed transitions, which we now use for prediction and training. The transitions are then processed as follows:

1. Receive sequence of actions from task state tracking (Section 5.4.2).
2. Associate actions to agent places and ensure that the sequence for each place is valid (i.e. pick action followed by corresponding place action).
3. Use enabled but not executed transitions for a marking as negative examples (obtained from goal-oriented transition filtering in Section 5.5).
4. Transform actions into the feature space (Section 6.3).
5. Train the DNN.

We now explain and motivate steps two and three in more detail. It is important to mention that the detection of an action does not necessarily represent ground truth data. Due to occlusion and sensor noise, incorrect actions might be detected. For example, when users move their hands over valid targets to place an object, an incorrect place action can sometimes be detected. Another source of errors is the structure of the Petri net. The user is modelled in terms of two agent places—one for each hand. This design decision is driven by two factors: (a) Users may use both hands simultaneously, and (b) the resolution of incorrect or incomplete observations requires an additional place to update the marking.

Three aspects must be taken care of to obtain a clean list of executed actions: Firstly, the exclusion of artefacts stemming from the update procedure. This means that when an object is temporarily observed as absent, this leads to the detection of a pick and a place action at the same location. Those instances are discarded in the action history. Secondly, the handling of action sequences of two hands. Thirdly, pick and place operations always take place alternately. For instance, a pick action can belong to a marking that is later discarded, because it no longer matches the observations. In that case, no corresponding place action will follow, and the pick action should not occur in the history. Multiple measures have been implemented to deal with the second and third aspects. Based on how many times each transition fires in the Monte-Carlo sampling, a weight in the range $[0, 1]$ is assigned. Transitions are recorded, and their weight added over several

updates. Only transitions exceeding a weight of 0.7 are added to the action history.Non-goal-oriented pick actions are only added if there is a corresponding place action for the same object type.

A major issue for training is that human preferences are not fully observable. That means, besides the action chosen, the robot does not know how much the user would have preferred or disliked taking other actions. Only over several assembly steps, the robot might learn other steps equally preferred by the human if the human executes them in a later run. A simplifying assumption is therefore that the user does not prefer the non-executed actions. Every added action thus generates multiple instances of training data. The action itself is a positive one (with a ground truth output of one). The remaining goal-oriented actions (as described in Section 5.5) form negative examples (outputting zero). Since positive training data is much less frequent than negative data, positive data must be given higher weight. Otherwise, the DNN output could converge to zero for all inputs. If there are n other goal-oriented actions for an executed action a, then a is given weight n. Practically, we replicate a n-times in the training data. This does not have a major impact on the size of the input data, as explained later. This concludes the description of how the input data is processed. Next, we briefly sketch the training process, followed by the processing of the DNN output.

Training and running the DNN is done in Caffe [189]. It was chosen for four reasons: (a) customizability and configurability, (b) capability of data indexing, (c) the direct C++ interface that allows reading input data of variable size from main memory, and (d) existing integration in the software stack to run the convolutional neural networks from Section 4.4. Data indexing refers to the procedure of providing data in terms of a data matrix and an index array. The entries in the index array tell the Caffe library which column of the data matrix to use. This reduces the memory footprint of the input data by a factor of around ten. Each training sample only requires five indices and the output value, instead of five 33-dimensional vectors. Whenever a new action is added to the action history and encoded into training samples, a new training step starts. Each training step consists of 1500 iterations. One iteration is the forward and backwards run with all training data, followed by a weight update. The training uses adaptive subgradient as a decay function because it is suited for sparse training data and does not require manual tuning of the learning rate [190]. The parameters are provided in Table 6.2.

A single training step with a hundred data entries only takes around a second when executed on the CPU. This makes the DNN suitable for online learning. Training data becomes available over time, and the DNN is updated in each step. The trained DNN is

Parameter	Value
Iterations per training step	1500
Learning rate	0.01, multiplied by a factor of 0.99 every 1000 iterations
Loss function	Euclidean $(\hat{y} - y)^2$, where y is the target output of 1 or 0 and $\hat{y}$ the predicted output by the DNN
Decay function	Adaptive Subgradient [191]

Table 6.2.: Parameters for training the DNN.

then available for the next prediction step. To avoid race conditions, a copy of the DNN exists that is only used for running the predictions. After a training step is completed, the weights are copied to the second DNN. The output of the prediction is a single scalar value. It resembles how likely the DNN thinks the candidate is executed. The score is then interpreted relative to the scores of the other candidates. Since the DNN requires outputting a global score for all actions, the scores of two actions can be rather close. Therefore, a post-processing is applied. Let w denote the DNN output for a specific action and W the multiset of the outputs for all action candidates. Then we update w as follows:

$$w' = \text{bell}\left(\max(W) - w, 0.3\right) \tag{6.2}$$

The updated values are afterwards normalised to form a probability distribution. This non-linear mapping of output values makes smaller differences between the high-ranked candidates more pronounced. In the case of two scores, no post-processing is applied. If the DNN has not yet seen enough training data, output values for all actions can be nearly identical, i.e. $\max(W) - \min(W) < 0.01$. In that case, we solely use the output of Algorithm 4 to get a rough approximation for preferences. However, we also use Algorithm 4 when the training data is very small. Let l denote the number of executed actions so far and w_2 the second largest element in W. If $\max(W) - w_2 \leq 0.25$, then add the weights from general assembly rules, but weighted with:

$$\min\left\{1 - \frac{1}{4}\left(\max(W) - w_2\right), 2^{4-l}\right\}$$

The weight factor exponentially decreases with the number of already observed actions and is small if the DNN states a clear preference for some action. This concludes the description of the intention prediction. We present a thorough evaluation in Section 7.5 after the data acquisition has been explained.

6.6. Conclusions and Discussion

This chapter has addressed the algorithmic replication of SMM in terms of predicting a probability distribution over the next action candidates given the previous actions. We reviewed action prediction approaches for human-robot teaming and for event processing (Section 6.1). In the former case, approaches utilise POMDPs, reinforcement learning, or stochastic models but do not scale for larger task scenarios. In the latter, state-of-the-art approaches merely utilise concurrent neural networks, which require a lot of training data. These approaches are unsuited for our requirements of incremental few-shot learning.

We therefore break down the prediction problem into approximating a weight function that gets as input the previous four actions and one candidate for the next action. Applying the weight function to all next action candidates gives a non-normalised distribution. The first contribution is a small DNN to approximate the weight function (Section 6.4). The DNN consists of just three fully connected layers. This comparatively small number of parameters makes it possible to train the DNN incrementally (Section 6.5).

The second contribution addresses the modelling of input data. In this regard, the main challenge is the incorporation of the state of the workspace in a condensed form. This is achieved by transforming each action plus the Petri net marking into a custom feature space. The feature space encompasses neighbourhood, position, colour, and volume information of the manipulated object (Section 6.3). The DNN has an extra layer to perform dimensionality reduction on the neighbourhood information.

Finally, the third contribution is the formalisation of the general assembly rules described by Mayer et al. [180]. They take the role of a fallback discriminator if the DNN does not have enough training data. Comparative studies have shown that these rules are prevalent across the human population but do not incorporate strategies on a fine-grained level (Section 6.2).

Our approach to action prediction could eventually be integrated into larger frameworks for modelling human behaviour. So far, reinforcement learning in simulation has proven to be a powerful tool for training robotic skills [192] and simple arm motions in human-like agents [193]. Looking ahead, more sophisticated simulations involving multiple agents are likely to become more common. A long-term vision is to proceed from action prediction to intention prediction. Automatic identification and prediction of human strategies could be a step towards truly foresightful and adaptive robot actions.

Due to design considerations, the proposed action prediction approach has several limitations: (i) Our approach is primarily targeted towards pick-and-place actions. These

are short-term actions with a well-defined start and end, that focus on a single object. Actions such as applying pressure, glueing, sanding, holding, or screwing are harder to encode into our feature space. When actions become more heterogeneous, object properties might no longer be sufficient to express all preferences. Instead, the feature space can be extended by tool usage, part usage, and the earliness of actions with high mental or physical effort [185].

(ii) Crafting the DNN involved many considerations to find a good balance between the number of parameters and the accuracy. However, it remains unclear whether each stage fulfils its intended purpose or is redundant or detrimental to the overall goal. Ablation studies [194] are one way of checking this. *Ablation studies* remove or degrade parts of a neural network to check the effect on the classification outcome. This gives insights into which layer is important for which discriminative aspect. In a future investigation, an ablation study can be used to thoroughly investigate how the network structure can be improved.

(iii) We currently need data from the interaction with a user to train the DNN initially. This can make the first encounter very counterintuitive. Ways to cope with this could be using the general assembly rules to generate synthetic data for the task at hand and pre-train the DNN. However, this can be counter-productive if personal preferences are the complete opposite. In that case, small-scale canonical task as proposed by Nemlekar et al. [185] can be executed in a microworld (cf. Section 6.2) to pre-train the DNN.

CHAPTER 7

Demonstrator and Evaluation

The central hypothesis of this work is that subjective fluency benefits from behavioural adaptation based on action prediction. Chapter 3 derived insights for human-robot teaming from human-human teaming. Chapters 4 and 5 laid the foundations for tracking human actions in a robust manner. Chapter 6 showed how to process the actions to train a model of human behaviour with incremental learning. The output of the model is a probability distribution over actions. After hand tracking has been evaluated in isolation in Section 4.6 and task state tracking in simulation in Section 5.7, this chapter benchmarks the components in a human-robot teaming setup. We start with the setup for the human-robot teaming (Section 7.1). Next, the coordination mechanism between the human and the robot for completing the task is introduced. This section focuses on the robots' decision-making processes (Section 7.2). Afterwards, we explain the study procedure, including the VR-based assistive support system for the human (Section 7.3). Interpretation of the results is then split into a comparison of the study conditions,

general correlations of fluency with other concepts, and a dedicated evaluation of the action prediction (Section 7.4).

7.1. Experimental Setup

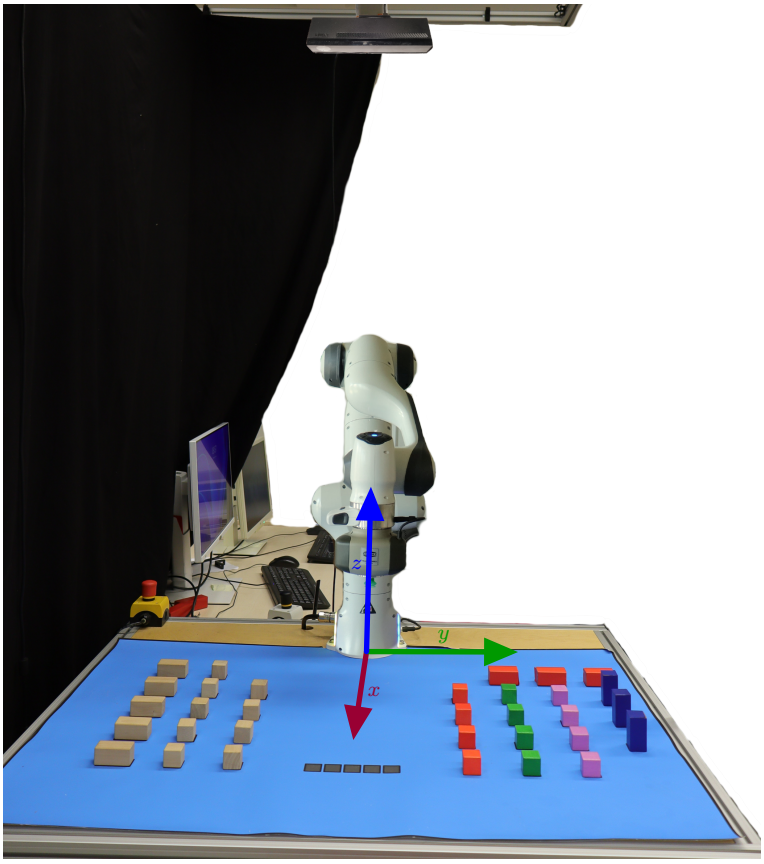
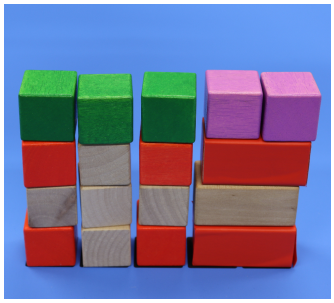


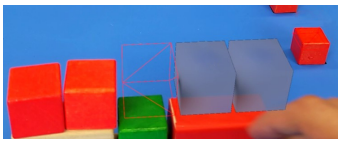
Figure 7.1.: Dimensioning, components, and configuration of the study setup. At the top, the Kinect 2 is mounted at a distance of $1.5\,\mathrm{m}$ from the table. Moreover, the workspace coordinate system is depicted, which is identical to the base coordinate system of the robot.

The hardware setup consists of a Franka Emika Panda robot equipped with a vacuum gripper, two computers, a Kinect 2, and a HoloLens 2. The arrangement of the hardware is shown in Figure 7.1. Precise hardware identifiers are listed in Appendix B.1. The software framework runs on a computer with an Intel i7-7700K core with four physical cores and a maximum clock rate of $4.50\,\mathrm{GHz}$, $32\,\mathrm{GB}$ of DDR4 RAM and Windows 10 installed. Additionally, a second computer is connected via Ethernet to receive motion commands and send control commands to the robot.

(a) TASK 1



(b) TASK 2



(c) View when wearing the HoloLens 2. The shaded cubes on the right show which ones still need to be placed. The red wireframe indicates where the robot place the next cube.

Figure 7.2.: Tasks and assistance system for the study.

Users sit in front of a table facing the robot. The collaborative working area is directly in front of them, whereas resources are located to their left and right. Task execution is supported with the Microsoft HoloLens 2 [80]. The HoloLens 2 displays the actions that still need to be done and the robot's next action (Figure 7.2c). The visualisation is anchored such that all virtual objects are displayed at their correct, real positions. This gives users a better immersion where resources need to be placed and where the robot performs its actions. The immersive plan is added to reduce errors by incorrectly placing blocks (Table 3.1).

The two structures depicted in Figure 7.2 are designed. Both have the same number of components, but differ in the number of blocks per layer. Both consist of a stack of larger blocks on the right that are well-suited for robot construction. If the human is not fast enough to complete the left part within time, the robot places blocks there, too. In that case, different task allocations might be optimal. Resources are picked from dedicated spots around the construction area (Figure B.2). The grey squares in the middle mark the *construction area*. The other coloured squares and rectangles around the construction area are the *resource pool*. The layout of the resource pool and the structures are manually coded in the software framework, and the Petri net is derived from it. The blue mat ensures that the user places the blocks in accordance with the positions coded in the software framework. The tasks are designed for the following five main goals:

(i) *Ease of use*: Users can execute the task without further training or advanced skills.

(ii) *Speed*: The robot can execute all actions with speeds in the same order of magnitude as humans. This ensures the robot can make some contribution to the task. Still, the robot does not operate at its speed limits so that users feel comfortable and safe while working in front of the robot.

(iii) *Habituation*: Tasks can be executed many times in sequence.

(iv) *Flexibility*: The task is characterised by both parallelism and dependencies to enforce proper task sharing.

(v) *Reproducibility*: Task components are readily available or easy to produce.

For performing pick-and-place actions, the robot executes point-to-point motions with intermediate points. An error-free action execution consists of moving into a start pose $10\,\mathrm{cm}$ vertically above the target location, approaching the target location, aspirating the object (or releasing it, respectively) and returning into the starting pose. To achieve an approximately linear motion when approaching the object (to avoid hitting other objects), an intermediate path point $3\,\mathrm{cm}$ above the aspiring pose is bypassed. An analytic inverse kinematics solver for the Franka Emika Panda [195] calculates the robot configuration given the pose of the tool centre point. All trajectories are point-to-point motions in joint space with a fixed maximum speed[1]. The robot requires around $4.5\,\mathrm{s}$ to complete an action. Between actions, transfer motions are necessary to reach the next starting pose. The robot then achieves around $0.35\,\mathrm{m\,s^{-1}}$—including starting and stopping.

7.2. Coordination

Previous studies investigating fluency used a machine-generated plan to coordinate human and robot actions (e.g. [21, 51, 196]). By contrast, this work emphasises the autonomy of the human decision-maker, who is free to choose his or her subsequent action. In consequence, human and robot need time to adapt to each other. Adaptation and familiarisation are achieved by repeatedly executing the same tasks. Both are favourable effects when investigating the concept of fluency. We therefore pursue an experiment design where participants execute the same task several times without the intervention of the experimenter, which resembles a standard scenario in small batch production where the same subcomponent assembly is completed multiple times in a row. To achieve repeated execution in a continuous run of a user study, the initial state of the setup must be restored. This is achieved by splitting the task into two phases, both to be carried out by the human and the robot together: composition and decomposition. In terms of the Petri net, the two phases share the same places, transitions and arcs. For each transition of the composition phase, a reverse transition for the decomposition

[1] We use a speed factor of 0.25 for the trajectory between intermediate path point and target location; and 0.6 for all other trajectories

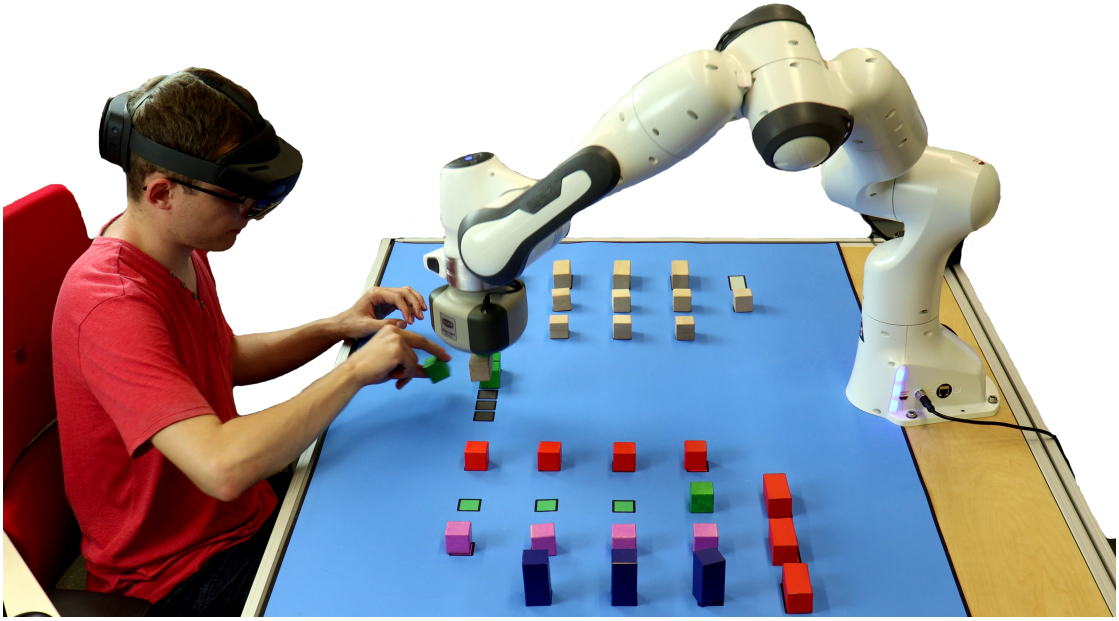


Figure 7.3.: Workspace and robot system. All objects are reachable by both human and robot.

phase exists. That way, the evolution of the workspace state can be tracked by firing transitions in the Petri net. No extra effort is required to obtain the initial state for the decomposition phase or to execute 'undo' actions. After completing a phase, only the set of goal instances needs to be updated.

Another challenge arising from decision autonomy granted to the human is that the robot needs to plan its actions dynamically based on previous human behaviour. Instead of generating a full plan of the whole task and performing a re-planning on every human action (e.g. [54, 49]), the robot system, as proposed in this thesis, only plans the next one or two actions. This keeps the planning overhead and delay at a few milliseconds in our setup. Action selection is guided by two system components. The first component is a set of deterministic rules and procedures that make a goal-oriented pre-selection of actions as described in Section 5.5. The second component is a conflict resolution system for equally feasible actions.

The first component is in place to ensure the robot does not get stuck and executes reasonable actions. To this end, an exploration of executable actions is based on the Petri net described in Section 5.3. The model uses separate transitions for pick and place actions and dedicated places to model the human's hand and the robot gripper. Since the vacuum gripper is quite bulky (Figure 7.3), some target place positions in the tasks cannot be reached, e.g. if there is a two-block-high tower next to them. To account for

that, a collision check is added to the condition in Algorithm 2 of Algorithm 2. The collision check calculates the release pose of the gripper for that place action. It then tests it against all placed objects (in the case of multiple hypotheses, these are all places that occur in at least one marking). The collision check uses the axis-aligned bounding boxes of the gripper and the placed object.

After the first system has identified equally feasible and executable actions, three robot behaviours are implemented to select the next action: FIXED, ADAPTIVE, and ADVERSARIAL. The FIXED strategy picks the first executable action according to the following order of preference: The structure is constructed layer by layer from right to left (from the user's perspective). Blocks are preferably picked close to the robot base. These rules are inspired by the assembly rules observed in human behaviour by Mayer et al. [180]. They are designed to avoid conflicts when starting the task. Most humans are assumed to start from left to right, picking objects close to them. Despite the name of the strategy, the robot does not necessarily select the same actions in every run of the assembly process. The FIXED strategy may select different actions in different runs when the action selection process is executed in different task states.

The ADAPTIVE and ADVERSARIAL strategies are based on the prediction process described in Chapter 6. To formalise those algorithms, we assume that we have a PREDICT function. The PREDICT function takes a set of transitions of the Petri net and optionally a list of predicted transitions as input. The function has access to the current belief marking and stores the previously executed actions. The list of previous and predicted actions is used as history when evaluating the DNN. The function then outputs the probability distribution described in Section 6.5 as a mapping from a transition to its probability value. Intuitively, the ADVERSARIAL strategy makes the robot select the same action that the user is assumed to carry out next, i.e. the one with the highest probability. Contrastingly, the ADAPTIVE strategy selects a supportive action. Supportive actions are those that minimise the risk that human and robot select the same action. Here, the assumption is that actions the user wants the robot to execute in a workspace state do not occur in the observed action history. The ADAPTIVE and ADVERSARIAL strategies mostly share the same planning algorithm. This planning algorithm distinguishes the following three major situations:

- **Composition**: Selecting a pick and place action when constructing the structure.
- **Decomposition**: Selecting a pick and place action when dismantling the structure.
- **Replanning**: Selecting a place action when the earmarked one can no longer be followed.

Algorithm 6: Prediction of Two Consecutive Actions

input : Given a coloured Petri net $N = (P, T, T_c, O, A, f)$, a set of goal instances $G \subseteq P \times O$ and a reachability tree S of belief markings.
output : Probability distribution P over $T_c \times T_c$

$Q \leftarrow$ `MaxPriorityQueue()`;
$P(t', t'') = 0 \; \forall t', t'' \in T_c$;
`Add(`Q`,` `Root(`S`),` 1`)`;
while *not* `IsEmpty(`Q`)` *and* $\sum_{t', t'' \in T_c} P(t', t'') < 0.9$ **do**
    $B, w \leftarrow$ `Pop(`Q`)`; // Returns element and weight
    if `Depth(`s`)` ≥ 2 **then**
        $t'' \leftarrow$ `LastAction(`B`)`;
        $t' \leftarrow$ `LastAction(Parent(`B`))`;
        Set $P(t', t'') = w$;
    $T_g \leftarrow$ `FilterGoalOrientedTransitions(`N, G, B`)`;
    $W \leftarrow$ `Predict(`T_g`,` `PreceedingActions(`B`))`;
    forall $t \in T_g$ **do**
        if $W(t) > 0$ **then**
            `Add(`Q`,` `GetChild(`B`,` t`)`, $w \cdot W(t)$`)`;
return P

The core component of **decomposition** is a tree search to explore the action space of the robot[2]. The root of the tree is the current state of the workspace encoded as a belief marking of the Petri net. Links represent actions executed by the robot, and child nodes represent future states of the workspace. Algorithm 6 shows how the probability distribution over action sequences is calculated. In the implementation, reachability is not explicitly constructed; only the paths followed are stored. If many possibilities for the first action exist, predicting the probability distribution for low-likely actions does not provide a benefit. Therefore, a threshold of 0.9 is used to save computational resources. If there is only one transition needed to reach the goal, the implementation uses a special case for Algorithm 6 to return the probability distribution over the next action only. Based on the result from Algorithm 6, ADVERSARIAL selects as actions for the robot the ones to

[2]The planning algorithm uses the transitions associated with the robot to explore the action space. To run the prediction, it does not make a difference whether the transitions is associated with human or robot because this information is erased when transitions are translated into the feature space described in Section 6.3.

be most likely executed next by the human:

$$a' = \arg\max_{t' \in T_c} \sum_{t''} P(t', t'') \text{ and } a'' = \arg\max_{t'' \in T_c} P(a', t''). \tag{7.1}$$

In contrast, ADAPTIVE selects the least likely:

$$a' = \arg\min_{t' \in T_c} \sum_{t''} P(t', t'') \text{ and } a'' = \arg\min_{t'' \in T_c} P(a', t''). \tag{7.2}$$

The minimum is restricted to those action sequences that are updated in Algorithm 6. The robot plans the next two actions so that it can communicate the next planned action to the user (Section 7.3). If several actions have the same probability, the order encoded in FIXED is used to select among them. The same applies to the beginning of the cooperation when no training data exists.

Algorithm 7: Prediction of Composition

input : Given a coloured Petri net $N = (P, T, T_c, O, A, f)$, a set of goal instances $G \subseteq P \times O$ and a reachability tree S of belief markings.

output : Probability distribution of place actions P'' over T_c and of pick actions P' over T_c conditioned on place actions in T_c

$T'_f \leftarrow$ `FilterGoalOrientedTransitions(`$N, G,$ `Root(`S`))`;
$\mathbf{B} \leftarrow$ `GetChildren(Root(`S`),` T'_f`)`;
$T''_f \leftarrow \bigcup_{B \in \mathbf{B}}$ `FilterGoalOrientedTransitions(`N, G, B`)`;
$P'' \leftarrow$ `Predict(`T''_f`,` `PreceedingActions(`$B \in \mathbf{B}$`))`; // use any pick action
$P'(t'|t'') = 0 \; \forall t', t'' \in T_c$;
forall $t'' \in T''_f$ **do**
  if $W''(t'') > 0$ **then**
    $\tilde{T}'_f \leftarrow \left\{ t' \in T'_f \;\middle|\; \text{out}(t') \subseteq \text{in}(t'') \right\}$; // pick actions suited for t''
    $W' \leftarrow$ `Predict(`$\tilde{T}'_f$`,` `PreceedingActions(`B`))`;
    forall $t' \in T'_f$ **do**
      Set $P'(t'|t'') = W'(t')$;
return P'', P'

The **composition** requires a different handling. Here, the constraint space around the structure requires some ahead-of-time planning to allow assembly in parallel. We therefore hypothesise that the user first chooses where to place a block on the structure and then where to pick it from. If Algorithm 6 were used, chances are high that the

block type that occurs most often in the workspace will be selected. To prevent that, the place action acts as the primary discriminator and the pick action is conditioned on the place action. Algorithm 7 outlines how the probability distributions are calculated. ADVERSARIAL then selects

$$a'' = \arg\max_{t'' \in T_c} P''(t'') \text{ and } a' = \arg\max_{t' \in T_c} P'\left(t' \mid a''\right). \tag{7.3}$$

ADAPTIVE selects

$$a'' = \arg\min_{t'' \in T_c} P''(t'') \text{ and } a' = \arg\min_{t' \in T_c} P'\left(t' \mid a''\right). \tag{7.4}$$

Finally, **replanning** is necessary if the planned action can no longer be executed. The check is run whenever the robot reaches the starting pose for an action (Section 7.1). An action is no longer executable if the user picked the object or placed one at the intended location. If that happens, replanning is executed: Algorithm 6 is run with the new belief marking to get a new distribution of forward transitions. If none exists, Algorithm 6 is run for all controllable transitions, including undo actions. A second problem arises when the suctioning fails due to misaligned objects. In that case, the action is aborted, and the action selection is rerun, but the failed action is excluded from the set of goal-oriented actions. If no action can be executed, the robot moves into a dedicated rest pose close to the base.

7.3. User Study Procedure and Methodology

The user study follows a hypothesis-driven mixed-model factorial design conducted in the laboratory. The between-subjects factor is the concrete task, and the within-subjects factor is the robot behaviour. This design is selected to (i) foster a direct comparison of robot behaviours, and (ii) to test the generalisability of the framework to different tasks and users. Established and validated questionnaires from the cognitive ergonomics domain are used to evaluate the research question Q5: The NASA-TLX [197] and the fluency questionnaire [21] (see Appendix B.2 for all materials). The combination of both covers a broad spectrum of cognitive ergonomics: cognitive workload, affect, satisfaction, subjective performance, personality, interaction quality, and trust [19].

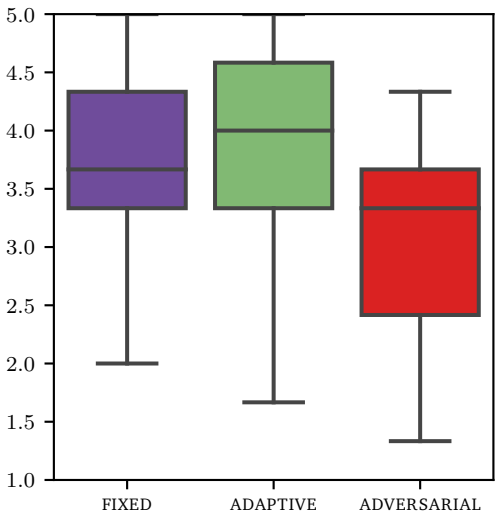
The NASA-TLX consists of six questions rated on a 100-points range with 5-point steps and a subsequent weighting of the questions. The fluency questionnaire consists of up to 27 questions grouped into eight subscales (Table 7.1) and rated on a 5-point Likert

scale. The fluency questionnaire is provided in both long and short versions. The short one consists of eight of the 27 items that are directly related to fluency and have high intercorrelation. Moreover, one of the subscales is named human-robot fluency, and all its items address fluency explicitly. The other subscales refer to downstream outcomes of fluency. To avoid confusion, this thesis annotates the term fluency with tags (long), (short), or (subscale) to refer to the version of the questionnaire or the subscale. The author of [21] notes that the choice of items for the short version still needs some fine-tuning. We report the reliability of the scales from the data of our experiment.

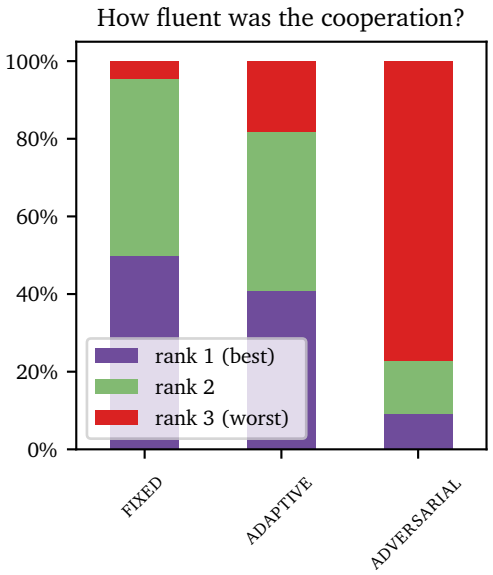
The procedure of the study was as follows: On arrival, participants read the task description (Appendix B.2), could ask questions and filled out the consent form. The experimenter supported the participants in putting on the HoloLens 2 and ensured that the virtual objects were properly visible and distinguishable. An introduction to the user interface of the HoloLens 2 was not conducted because participants could not directly interact with or modify the virtual environment. To familiarise themselves with the task and get used to the HoloLens 2, participants constructed and dismantled the structure once without the robot[3]. The participants were then exposed to the three robot behaviours in a random order, which, from now on, are denoted as *trials*. The experimenter announced the beginning and end of each trial. Each trial lasted for approximately ten minutes to collect sufficient training data and to foster familiarisation and adaptation effects. Participants managed to construct and dismantle the structure between four and ten times, depending on their individual working speed. At the beginning of each trial, the experimenter reset the robot's behavioural model of the human. This was done to reduce confounds from the previous trial. While the robot was moving, the experimenter had a grip on the emergency stop to ensure the physical safety of participants. After each trial, the participants filled out the questionnaires at a computer next to the robot. After the last trial, participants additionally filled out the final questionnaire (Appendix B.2). This ended the study, and the experimenter informed the participants about the robot behaviours and study setup.

Participants were recruited through notices on campus and mail. They were then allocated a timeslot. Participants did not receive monetary compensation. In total, 24 persons participated in the study. Three were female and 19 were male. Two did not specify a gender. The average age was $24\,\mathrm{years}$. All had normal or corrected-to-normal sight. None stated to have sensorimotor impairments that could affect the study. Excluding outliers, participants took from $88\,\mathrm{s}$ to $170\,\mathrm{s}$ to construct and dismantle the

[3] They could do more test runs on request



(a) Fluency (subscale) for each behaviour, rated at the end of each trial. The subscale is composed of three items rated on a 5-point Likert scale from "strongly disagree" (1) to "strongly agree" (5).



(b) Participants ranked all three behaviours at the end of the study.

Figure 7.4.: Rating of fluency after each trial and at the end of the study.

structure (hereinafter referred to as a *run*). All participants successfully managed to complete multiple runs per trial. It happened from time to time that the vacuum gripper did not successfully suck in a block—mostly due to imprecisions of the end effector. In those cases, the robot noticed the failed attempt and took the next one. In rare cases, the structure became unstable or partly collapsed. Participants were then instructed to continue with the task by fixing the broken part.

7.4. Study Results

The analysis is split into three parts to address the research questions Q4 to Q6. The first part addresses how fluency is impacted when the robot's decision-making is based on predictions of the human's behaviour (Section 7.4.1). The second part links the findings to subjective and objective productivity metrics (Section 7.4.2). Finally, the last part provides an in-depth quantitative analysis of the prediction algorithm (Section 7.5).

Subscale	Cronbach's Alpha	Confidence Interval
fluency (short)	0.854	[0.801, 0.899]
human-robot fluency (subscale)	0.709	[0.578, 0.805]
robot relative contribution	0.770	[0.675, 0.843]
trust in the robot	0.688	[0.547, 0.791]
positive teammate traits	0.796	[0.712, 0.861]
improvement	0.750	[0.637, 0.833]
working alliance	0.812	[0.741, 0.869]
goal perception	0.806	[0.719, 0.870]

Table 7.1.: Subscales of the fluency questionnaire [21] with Cronbach's α on a 95% confidence level.

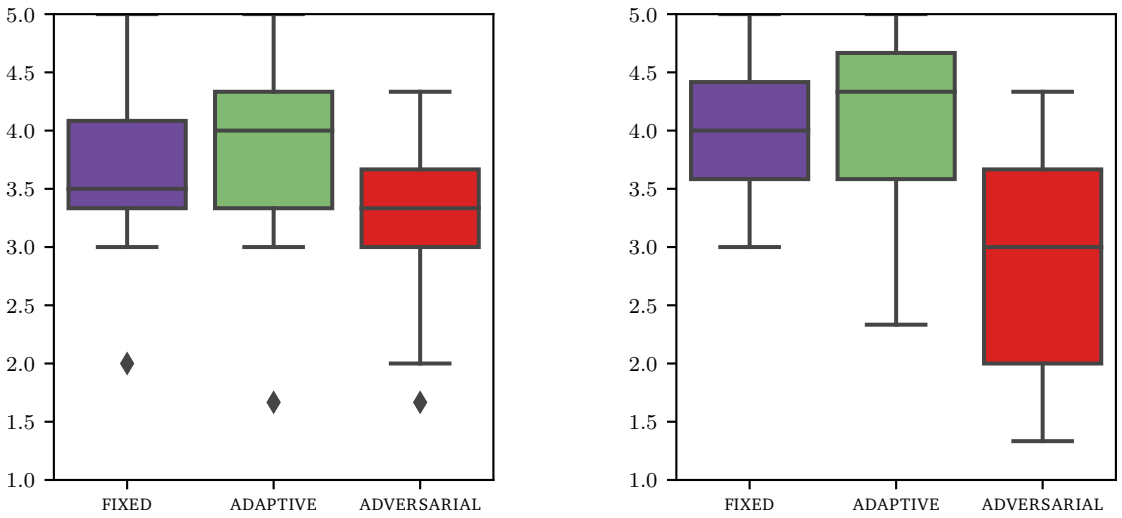
7.4.1. Fluency Metrics

The foundation for the evaluation is the human-robot fluency questionnaire by Hoffman [21]. Question items are assigned a numerical value between one and five and are arithmetically averaged to obtain a rating for each subscale. A reliability analysis shows that all subscales except trust in the robot have acceptable to good reliability (Table 7.1). Reliability values are similar to those in the literature, except for human-robot fluency (subscale) and trust in the robot, which are far less reliable. Possible reasons are that most participants got the German version of the questionnaire and interpreted the improvement item differently.

Next, a Shapiro-Wilk test is run on each subscale for each behaviour to test whether the data follows a normal distribution. The majority of distributions give a p-value below 0.1, so that non-parametric estimators and tests are chosen for all following tests. A Friedman Test on a confidence level of $95\,\%$ confirms that the robot behaviour has a significant effect on all subscales except for robot relative contribution and goal perception. We further analyse the differences with pairwise comparisons. We use a one-sided Wilcoxon signed-rank test to test significance and the Hodges-Lehmann-Sen shift estimator to obtain the shift of medians. All tests are conducted on a Bonferroni corrected confidence level of $98.3\,\%$ (cf. Appendix B.3.1). The null hypotheses are that

- ADVERSARIAL is better than or equal to FIXED and
- FIXED better than or equal to ADAPTIVE in each subscale.

Figure 7.4a depicts the participants' ratings regarding the dimension fluency (subscale). Both the FIXED and ADAPTIVE strategies result in a similar mean value. Both were perceived as fluent in general. Contrary to the expectation, ADAPTIVE is not perceived as



(a) Perceived fluency of TASK 1.

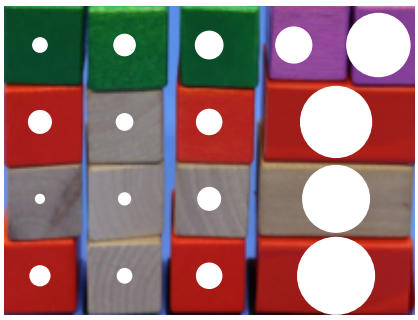
(b) Perceived fluency of TASK 2.

Figure 7.5.: Comparison of the two tasks with regard to fluency (subscale). Higher values are better.

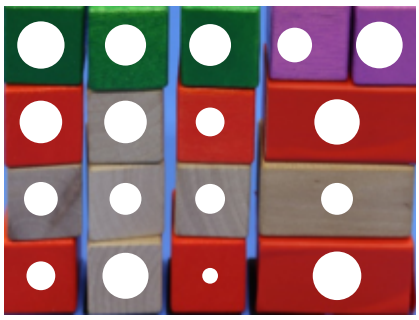
significantly more fluent. One reason is that the ADAPTIVE strategy is sensitive to the first runs. If a participant decides to grasp blocks far away to speed up the cooperation, the robot learns the behaviour and assumes that the human prefers those. Other reasons to explain the high variance in the ratings are the robot's grasp failures and the overall unfamiliar situation. In the free form responses, participants approximately equally often stated that the robot followed their preferences or preferred the blocks further away for the FIXED and ADAPTIVE strategy. However, the ADAPTIVE strategy was described as a bit random in the beginning by three participants. This highlights the issue that mutual adaptation can lead to volatile schedules.

A closer look into the results reveals that TASK 2 was perceived as slightly more fluent than TASK 1 (Figure 7.5)—though the confidence interval of the Hodges-Lehmann-Sen estimator overlaps with zero, i.e. the shift is non-significant. A possible reason is that participants achieved a better action allocation in TASK 2 as compared to TASK 1. Inspecting which blocks were placed most frequently underlines this hypothesis: For the FIXED strategy, the robot mostly placed the large blocks in the right part of the structure (Figure 7.6) for both tasks. However, the robot's place actions for the FIXED strategy in TASK 1 are more scattered.

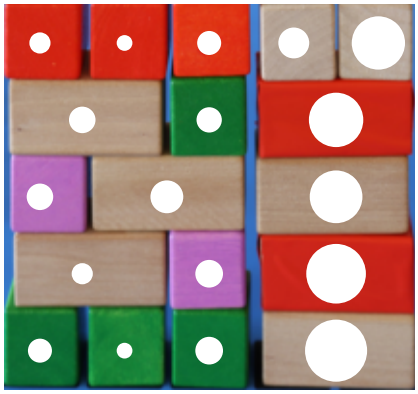
Comparing the FIXED and ADVERSARIAL strategy, however, yields a strong and significant difference of $0.6\,\text{points}$ (confidence interval: $[0.3, 1.3]$) in favour of the FIXED strategy.



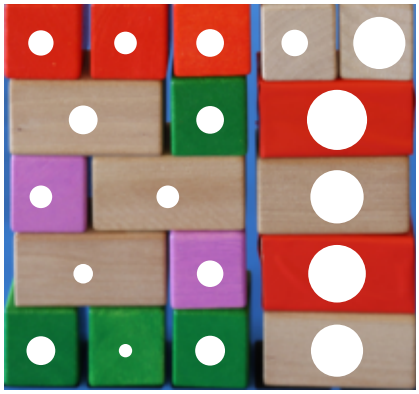
(a) FIXED on TASK 1



(b) ADAPTIVE on TASK 1



(c) FIXED on TASK 2



(d) ADAPTIVE on TASK 2

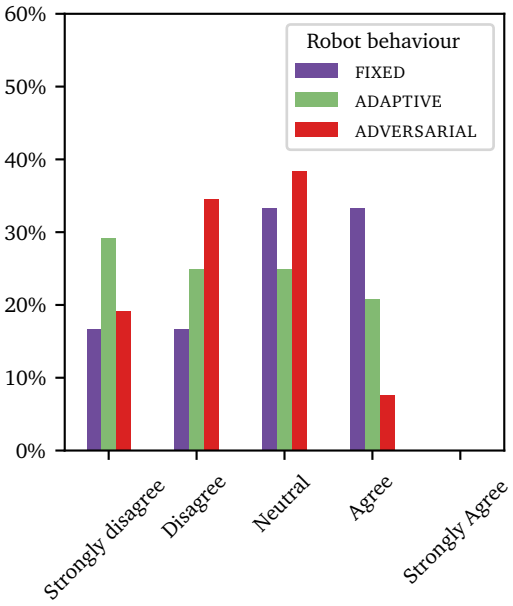
Figure 7.6.: The area of the white circles indicates how often the robot placed this block. Data is aggregated over all participants and runs.

The ranking of the strategies supports the observations. Except for three, all participants ranked the ADVERSARIAL as least fluent. They stated that the robot often picked blocks close to them from the resource pool. The robot was thus capable of learning and mimicking the participant's behaviour to an extent that they felt annoyed by. This provides a first insight into the accuracy of the learning algorithm. We further elaborate on its accuracy and performance in Section 7.5.

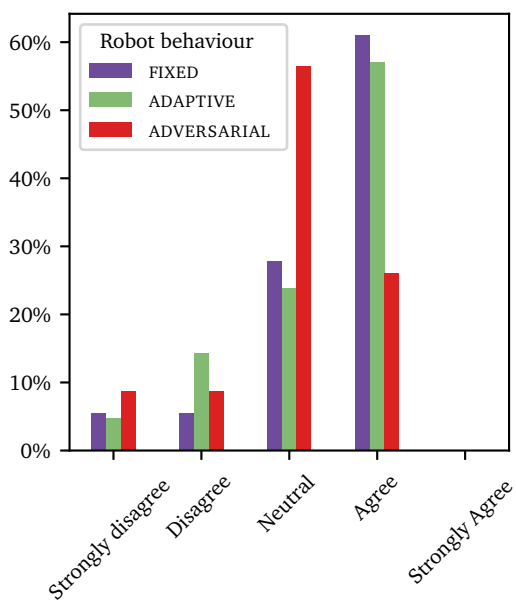
The trend is consistent when it comes to positive teammate traits, improvement, working alliance, and subjective performance. All show a significant and strong effect. The relative contribution of the robot, trust in the robot, and goal perception do not show a significant effect. The robot's contribution to the task is limited by the speed at which it can pick up and place cubes. To ensure accurate performance, the speed had to be set to a limit far lower than what humans can achieve. Participants thus rated the robot's contribution overall as low. In contrast, goal perception was rated fairly high for all behaviours, but with a large variance. Participants do not seem to experience a large difference in how the robot perceives their goals as long as it continues to assemble the structure. Except for working alliance and improvement, the comparison of ADAPTIVE and ADVERSARIAL yields the same significance and strengths of effect as the comparison of FIXED and ADVERSARIAL. In contrast to the expectation, however, participants do not notice a significant improvement when comparing the FIXED with the ADAPTIVE strategy. As depicted in Figure 7.7, the team performance improves for the FIXED and ADAPTIVE strategies, but the robot's performance is perceived as degrading on average. The observation even holds for the FIXED strategy despite the robot not being programmed to show any difference in behaviour. A shortcoming here is the interpretation of the questionnaire items by the participants. Two participants stated that they ticked 'strongly disagree' because the collaboration was fluent right from the beginning. A differential scale, reworking the formulation, or a 'not applicable' option might be necessary here to prevent participants from coming up with their own interpretation, which can be inconsistent among participants [198].

The significant differences are no longer present when using the NASA-TLX as a scale (Figure 7.8a). The plot shows a slight, yet insignificant, shift of means. Only the effort sub-scale, which measures physical and mental effort to accomplish the performance, shows a significant difference ($p < 0.014$) between FIXED and ADVERSARIAL (Figure 7.8b). Overall, the indifference of NASA-TLX shows that subtle differences, e.g. in terms of fluency, are not necessarily captured by an overarching measure.

The robot's performance improved over time.



The human-robot team improved over time.



The human-robot team's fluency improved over time.

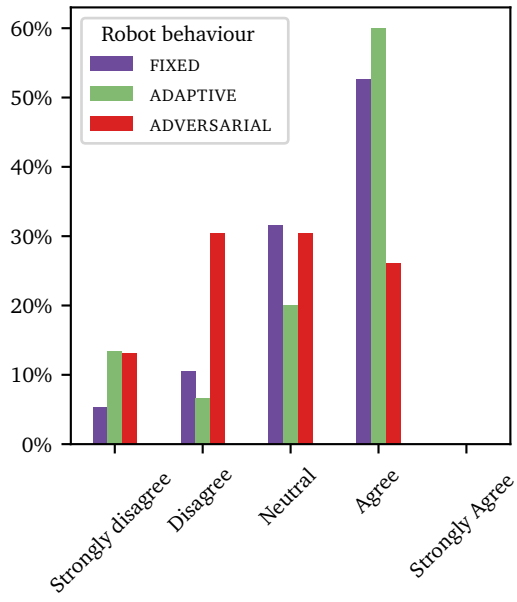
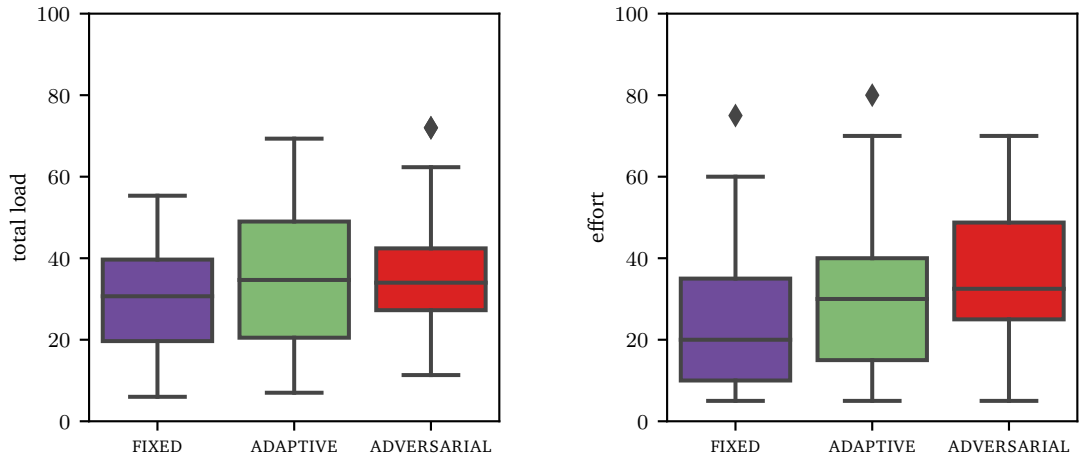


Figure 7.7.: Responses to the individual items that constitute the improvement subscale.



(a) Combined and weighted NASA-TLX.

(b) Effort sub-scale of the NASA-TLX.

Figure 7.8.: Ratings of the robot behaviours for the weighted NASA-TLX and a subscale thereof. Lower values are better.

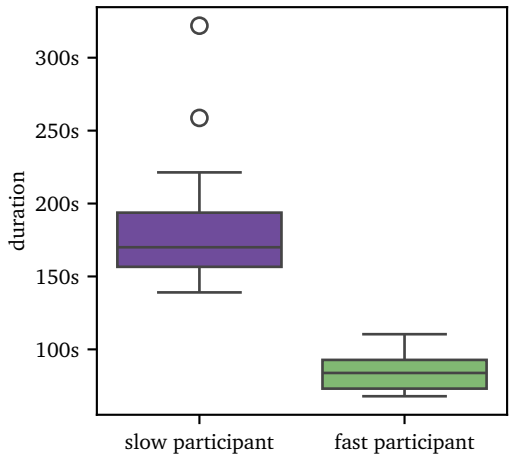
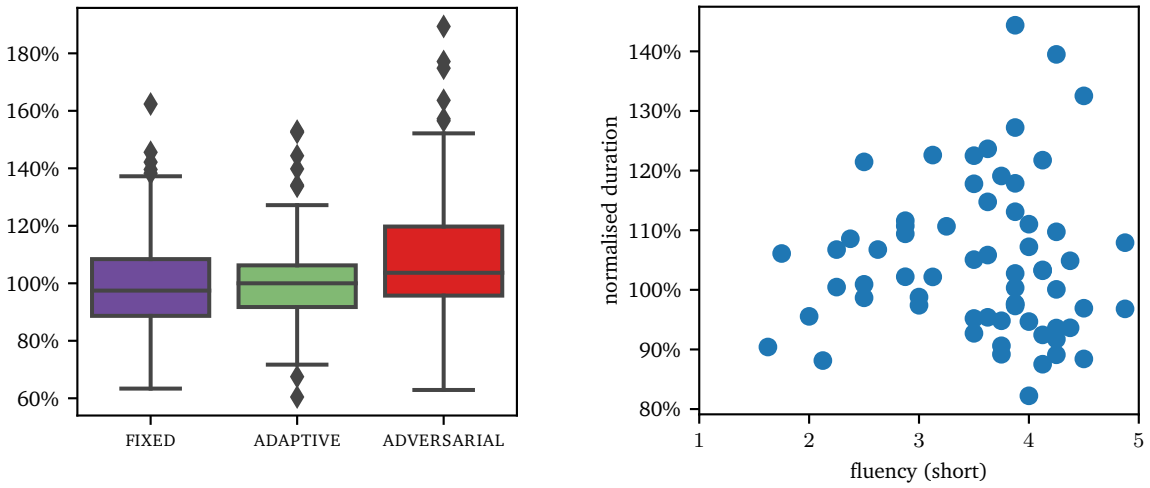


Figure 7.9.: Durations of one run by the slowest and fastest participant. Each data point denotes the duration of assembly and decomposition of the structure.



(a) Variances of duration around the median duration (100 %) of each participant.

(b) The short fluency scale plotted against the participant's normalised duration. Since the normalised duration is calculated per run, these values are averaged to get one value per trial.

Figure 7.10.: Investigation of normalised duration.

7.4.2. Correlations Between Metrics

Since we have found a strong correlation between the robot behaviour and fluency in the previous section, we now investigate how metrics from cognitive ergonomics correlate with objective productivity metrics in an exploratory factor analysis. Common metrics to measure productivity are human-idle time, robot-idle time, concurrent activity, and completion time per assembly step [19]. The term *activity* refers to the amount of time an agent actively contributes to the task, i.e. the agent performs an action. In contrast, *idling* is the time the agent waits, e.g. for the other agent to complete a prerequisite action.

Task times are obtained from the robot's data logs. Figure 7.9 shows an example of task durations for two participants. Working speeds differ by a factor of two. Previous work investigating assembly tasks with unskilled participants has also found a high variance for execution times among participants [185]. The authors attribute this to the different skill levels.

We thus have a systematic shift of means between different participants that massively inflates the variance if we group the data points by robot behaviour. To account for that, we divide durations by the user's median duration. We therefore define 'fast' and 'slow' task executions of a certain participant as deviations from the median. Figure 7.10a

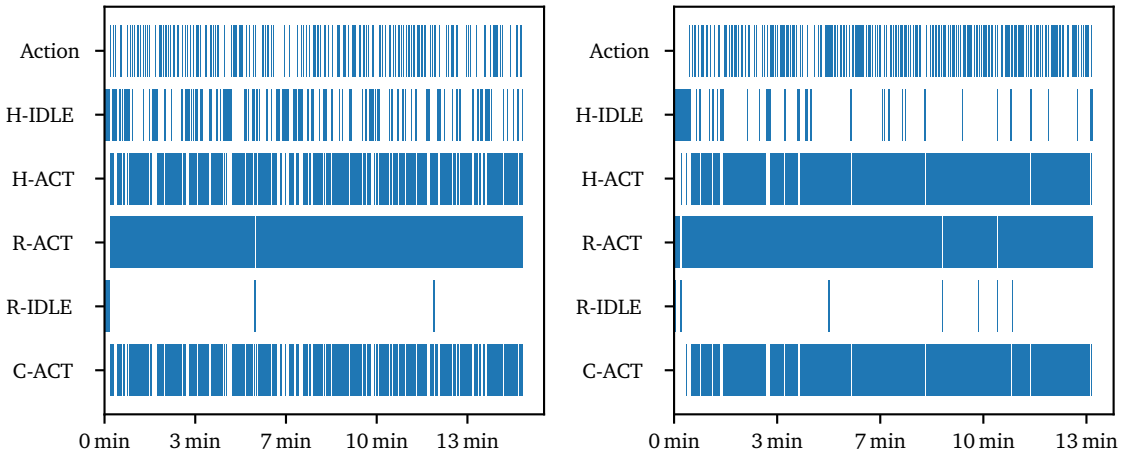


Figure 7.11.: Gantt charts of a slow worker (left, 5 runs completed) and a fast worker (right, 10 runs completed). Depicted measures are human idle (H-IDLE), robot idle (R-IDLE), human working (H-ACT), robot working (R-ACT), concurrent activity (C-ACT), and detection of a completed action (Action).

shows that participants' normalised task durations show the same significant differences ($p < 0.001$) when comparing the ADVERSARIAL behaviour with both others. However, the fluency and normalised duration are not proportional to each other, as shown in Figure 7.10b. We thus further explore what other factors are correlated with fluency.

Besides task durations, the relation of human and robot activity is an important indicator for the productivity of a human-robot team [19, 21, 199]. Robot activity can be directly extracted from the logging data. Extracting human activity requires a more sophisticated approach. Other studies from the literature use confirmation buttons to track the start and end of an action (e.g. [51, 200, 201, 54, 202]). Our study is more flexible and task-centred. The human can choose which action to perform next. There is no distraction by confirming task completion to the robot system. The robot tracks the state of the workspace to detect completed actions. However, due to occlusions the detection may happen several seconds after the human completed an action. Moreover, tracking cannot detect when the human started an action.

To overcome the lack of direct logging data, the hand movement is used as a proxy for human activity. The workspace is split into three zones along the x-axis. The activity of the human is calculated based on the hands being in one of the zones and their velocity. Further details and parameters of the implementation are explained in Appendix B.3.2. In total, 71 log files were processed, each containing the hand poses of one participant in one trial. Three trials were dropped due to incomplete logging data. Two sample outputs

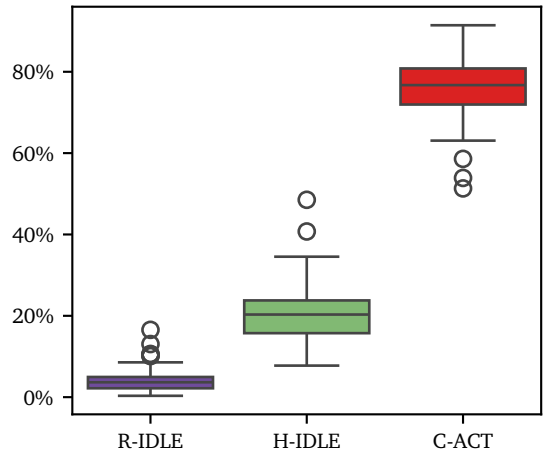
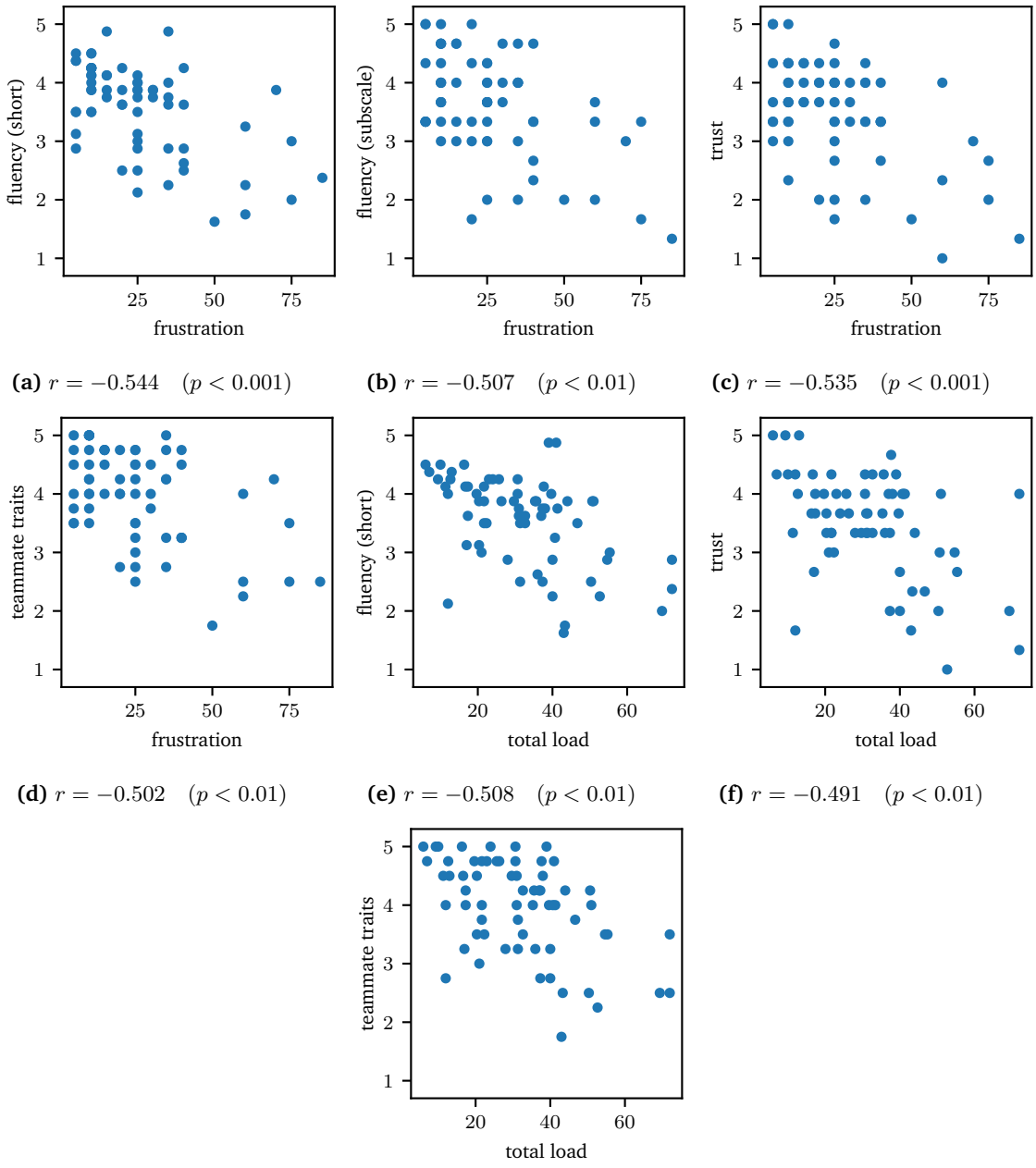


Figure 7.12.: Percentage of human idle (H-IDLE), robot idle (R-IDLE), and concurrent activity (C-ACT) per participant and trial.

are depicted in Figure 7.11. Times of activity are put into relation with the overall task duration and robot activity to obtain the percentage of idle time and concurrent activity. Figure 7.12 shows their distribution. The robot was continuously and consistently active. The robot only moved into the resting position in rare cases where no action was possible. Human idle time has a larger variance. The data points at the lower end show that the tasks can be organised in a way that makes humans almost always active. However, some participants preferred to wait for the robot to communicate its intention or complete the action before starting their own.

The Pearson correlation coefficient is used to quantify the linear correlation between productivity indicators and question items from the short fluency questionnaire. Significance is tested with a two-tailed test and a threshold of $p < 0.05$ to match the comparison in Hoffman [21][4]. Results are summarised in Table 7.2. Only human idle time and robot contribution yield a significant correlation. In line with the online fluency study results by Hoffman [21], the correlation is positive, i.e., the robot contributes more to the success of the team if the human is more idle. However, other pairs are not significantly correlated. Given the low correlation in the online fluency study, where participants could not interact with the environment but watched and rated videos, this indicates that the effect might be too weak to be detectable in a real-world cooperation scenario without controlled human idle times.

[4]Functional delay cannot be investigated since the human cannot directly instruct the robot.



(a) $r = -0.544 \quad (p < 0.001)$ **(b)** $r = -0.507 \quad (p < 0.01)$ **(c)** $r = -0.535 \quad (p < 0.001)$

(d) $r = -0.502 \quad (p < 0.01)$ **(e)** $r = -0.508 \quad (p < 0.01)$ **(f)** $r = -0.491 \quad (p < 0.01)$

(g) $r = -0.503 \quad (p < 0.01)$

Figure 7.13.: Significant correlation between productivity and fluency metrics. The x-axis is a subscale of the NASA-TLX rated on an interval from 0 to 100, and the y-axis is a subscale score of the fluency questionnaire rated on a 5-point Likert scale, ranging from strongly disagree (1) to strongly agree (5). The sub-captions show Pearson's r and the Holm-Bonferroni corrected p-value.

	R-IDLE	H-IDLE	C-ACT
The human-robot team worked fluently together.	.049	.090	-.094
I was the most important member on the team.	-.117	.098	-.057
The robot was intelligent.	-.220	.052	.018
The robot was trustworthy.	-.055	.022	.010
The robot was cooperative.	-.155	.056	-.008
The robot contributed to the fluency of the interaction.	-.244	-.052	.124
The robot was committed to the success of the team.	-.184	.070	-.014
The robot had an important contribution to the team.	-.100	**.275***	-.225
Fluency (short)	-.188	.105	-.039

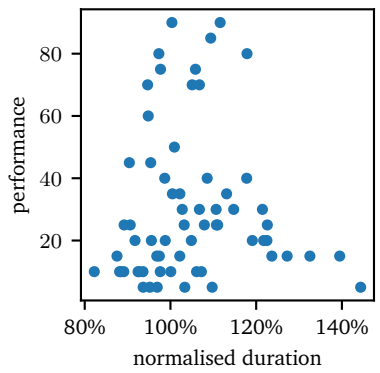
Table 7.2.: Correlation between indicators of objective productivity and subjective fluency. $*p < 0.05$

We further explore the relationship between indicators of performance, workload and fluency. Pairwise correlation tests with Pearson's r are conducted. To avoid an inflation of Type I error, a Holm–Bonferroni correction is applied to the p-values. Of the 108 pairwise tests, seven were significant. Figure 7.13 shows the significant correlations. Results are accompanied by a scatter plot to ensure they form a bivariate normal distribution, which is a requirement for Pearson's r.

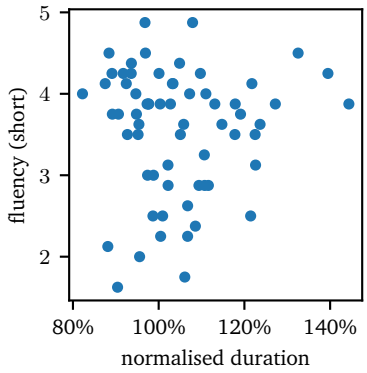
Frustration and total load are both strongly negatively correlated with fluency (short), trust, and teammate traits. On the one hand, the correlations show that total load and fluency have a common ground. Frustration seems to be one of the most sensitive subscales of NASA-TLX when fluency changes. On the other hand, the short fluency questionnaire promises to be an effective instrument for identifying factors for total load. The correlation of trust and teammate traits is in line with previous research [203, 204, 57, 205] that identifies these as important factors for successful cooperation.

It is important to mention that further research is required to identify causal links between these factors. Moreover, fluency, trust and teammate traits still need to prove their predictive capabilities with respect to total load and frustration in future studies.

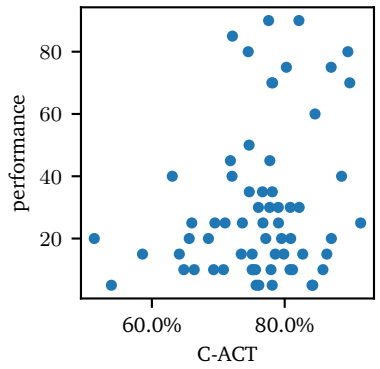
When inspecting objective performance indicators, subjective performance, and fluency indicators, we do not observe significant correlations as depicted in Figure 7.14. This supports the claim (e.g. by [19, 21, 206]) that fluency and subjective performance are not just redundant to objective performance indicators but can give additional insights.



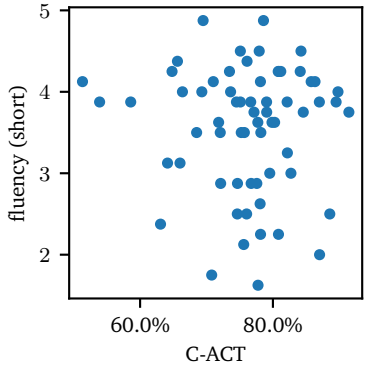
(a) normalised duration vs. performance (NASA-TLX)



(b) normalised duration vs. fluency (short)



(c) concurrent activity vs. performance



(d) concurrent activity vs. fluency (short)

Figure 7.14.: Comparison of productivity and subjective performance-related indicators. Neither of the depicted correlations shows a trend.

7.5. Evaluation of Action Prediction

In this section, we use the logging data from the study to evaluate the precision of the DNN and compare it with a prediction method based on Markov chains [107]. The logging data consists among other of all emissions (Section 5.4.1), belief markings (Section 5.4.2), hand trajectories (Section 4.6), robot actions (Section 7.2), robot poses, human actions, the prediction results (Section 6.5), and the binary training data and weights of the DNN. All log entries have a global timestamp to associate events and states by time. Since the prediction only runs when the robot plans its next action, only a few predictions per run are logged. To get a better picture of the overall performance, we conduct a thorough analysis outside the software framework. Therefore, the Python interface of the Caffe library is used, and the algorithms from Chapter 6 are re-implemented in Python. The re-implementation includes a dedicated class model to parse and store the logging data. To keep the implementation effort manageable, all other parts of the software framework, such as the Petri net and reasoning components of Section 5.3, were not re-implemented. The evaluation of prediction was restricted to the next action label. The anticipatory approaches from Algorithms 6 and 7 were not implemented due to the tight coupling with the Petri net.

A random manual check of the logged actions revealed that they suffer from errors and inconsistencies. Errors occur when the task state tracking compensates for measurement errors. This can happen, for instance, when the human hand intersects a place. The task state tracking then adds an action that places a block there. Once the human hand leaves the construction area, the task state tracking creates undo actions to match the state of the workspace. Inconsistencies happen when the belief marking has to track multiple concurrent actions. These are then not added to the history since their probabilities are too low. Later, when the uncertainty is resolved, belief markings are discarded, but we do not track which of the previous actions was actually executed. In consequence, the logged action can contain consecutive actions of the same type and hand, or pick-and-place sequences where the grabbed object type does not match. We therefore apply the following data **preparation procedure** to obtain action sequences with fewer errors:

Manipulation Detection The logged belief markings are processed to identify events where tokens are removed or added to places. At each timestamp, only the marking with the highest probability is considered. Pick and place actions are created if the token moves from/to the user's hand.

Hand identification Based on the hand trajectories, the used hand is identified. Criterion is the smallest distance within a one-second window before the event.

Filtering Within each run and per hand, actions are removed if they meet the following requirements: (i) An action at the same place with the same token type was executed before, then remove both. (ii) The previous action has the same type.

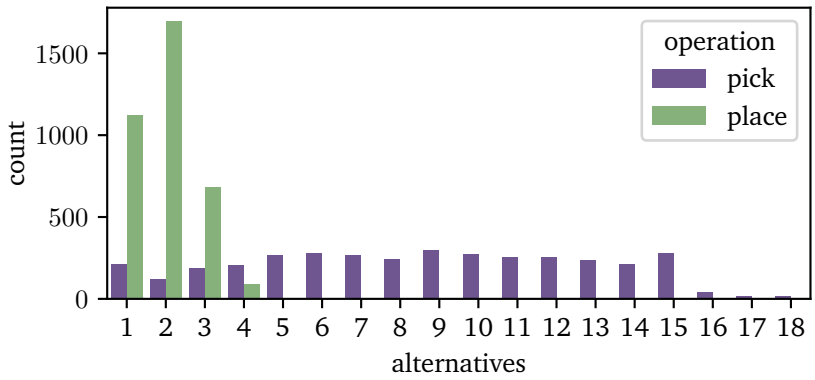
Augmentation For each remaining action in the run, a list of alternative actions is added. To obtain the list, a command line interface is added to the main software framework that gets one belief marking from the logging data and outputs the goal-oriented actions (Algorithm 2).

Splitting Runs with robot behaviour ADVERSARIAL are removed, all others are split into the datasets D-DECOMP for the decomposition of any of the structures, D-TASK1 for the construction of TASK 1, and D-TASK2 for the construction of TASK 2. The individual datasets are used to identify individual strengths and weaknesses. For the comparison, the datasets are combined.

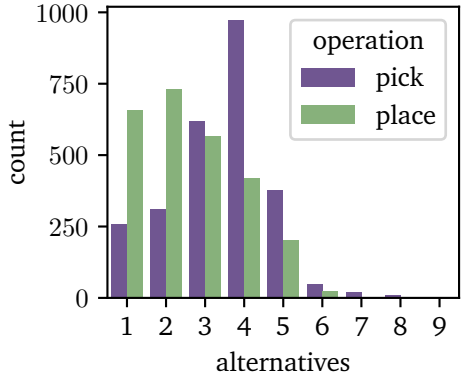
The prediction problem is treated as a multi-class classification problem. Class labels are all locations of the workspace (as specified by the task model). However, at each timestamp, there is only a certain set of actions that can be executed. The classification can therefore be restricted to this subset of class labels (see Figure 7.15b and Figure 7.15a for the distribution of the sizes of these subsets). Moreover, the classifications are not independent but require previous actions as input. Therefore, we must process a complete run at once. We use four classification algorithms and two ways of splitting the training and test sets (referred to as *validation schemes*). The two **validation schemes** for each dataset are:

Cross-Validation This corresponds to a 24-fold cross-validation (because of 24 participants). One participant serves as validation. All runs of that participant form the validation data, and all other runs form the training data. Since each participant is used once for validation, we get one evaluation result per participant and classification algorithm.

Within-Participant Validation The dataset is limited to one participant. From the runs $20\,\%$ are used as validation and the others for training. This results in two to three runs for validation, depending on the total number of runs in that trial. We create 33 splits of training and validation data per participant and thus get 33 evaluation



(a) Count of actions grouped by alternatives for D-TASK1 and D-TASK2 combined. Alternatives '1' means there is only 1 action to execute.



(b) Count of actions grouped by alternatives for D-DECOMP.

dataset	runs	actions
D-DECOMP	274	226
D-TASK1	193	271
D-TASK2	130	255

(c) Number of runs and average number of actions per run in each dataset.

Figure 7.15.: Statistics of the datasets used for training and evaluation.

results per participant for each algorithm. The splits are created pseudo-randomly so that each algorithm can be evaluated on the same set of splits.

The term *evaluation result* refers to a table where each row represents the output of one classification algorithm. All rows in an evaluation result stem from the same classification algorithm with the same training data. Columns of the evaluation result are the executed action (correct label), the action with the highest prediction score (predicted label), the highest prediction score (prediction score), and the number of alternatives. We use the following classification algorithms:

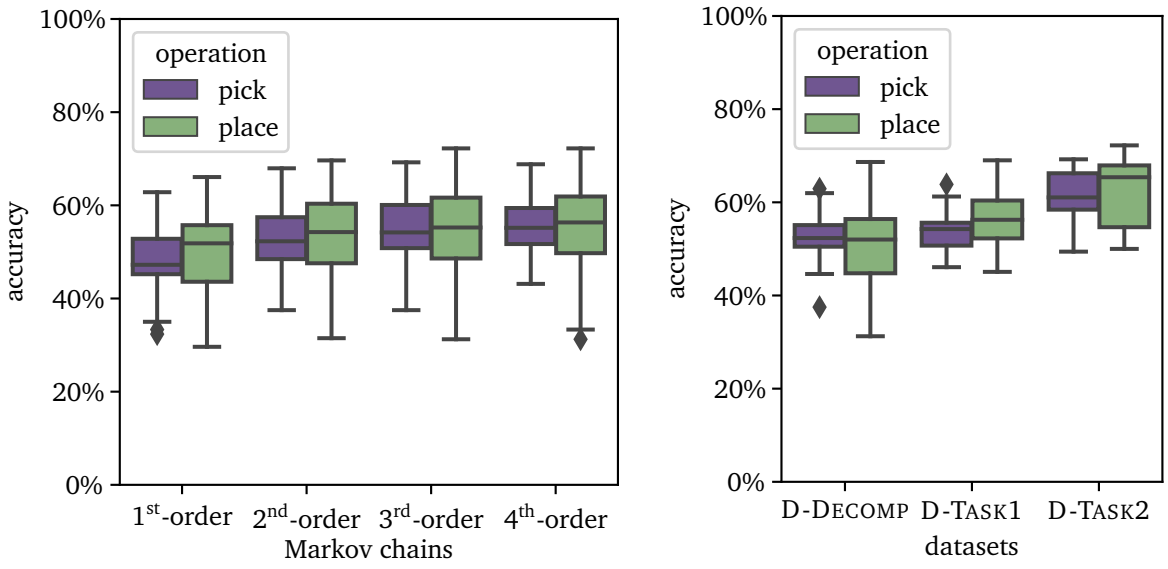
Random One action of the alternatives is picked at random. This serves as a baseline to show how much the other classifications are better than guessing. Since the number of alternatives varies, this baseline shows the difficulty of the prediction problem.

Markov Chains This is a direct implementation of Zanchettin et al. [107]. A first-order Markov chain is a transition probability matrix from one action to the next. A row represents the probability distribution of the next action given the previous one. A Markov chain of order n is approximated by a weighted sum of n transition matrices. Each matrix represents the transition from the i-th previous action to the next action. At first, the training process initialises all transition matrices. Then, the weights for the sum are calculated to minimise the prediction error. For the evaluation, we filter the row by the goal-oriented, executable actions at that time step. The resulting distribution is then scaled to a probability distribution.

Assembly Rules Applying the assembly rules from Section 6.2. The algorithm is called twice with the two previous actions as a_0. The results are combined by summing the weights.

DNN The DNN-based procedure presented in Section 6.5. Since there is sufficient training data before starting the evaluation, the fallback solution of using the assembly rules hardly plays a role. Only the first three actions of each evaluation use the fallback.

For plotting, pick and place actions are treated separately. We calculate accuracy per evaluation result as the ratio of correct predictions divided by total events. Events with just one alternative are ignored. The accuracy of each evaluation result is therefore one data point in the following plots. The accuracies per validation scheme, classification algorithm, dataset, and operation were plotted into normal quantile plots. All plots



(a) Combined accuracy of the Markov chains for all datasets.

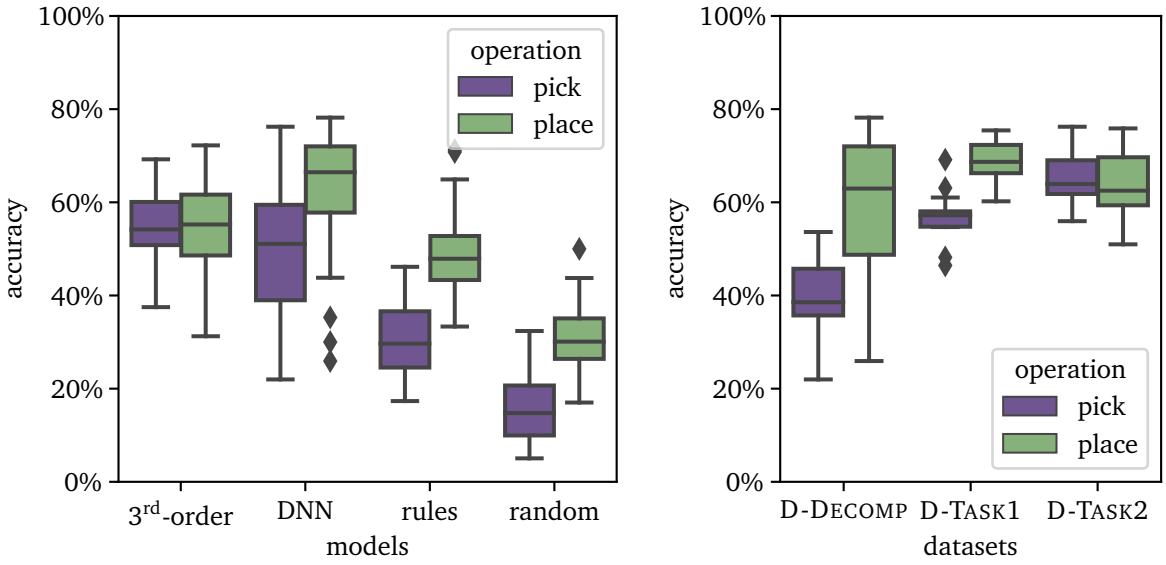
(b) Accuracy of the **3rd-order Markov chain** for different datasets.

Figure 7.16.: Accuracy of Markov chains with **cross-validation**. Higher values are better.

show that the distribution of the accuracies closely follows a normal distribution. In the following, we use paired t-tests to test for significance.

We start by investigating the cross-validation to see fine-grained differences between models. Since Markov chains have order as their hyperparameter, we first pick one representative for that model before comparing the models with each other. Figure 7.16a depicts the comparison among the Markov chains only. Increasing model complexity moderately increases prediction accuracy by up to 7 p.p. However, improvement stagnates with the third order. The difference between third and fourth order is no longer significant. In contrast, the comparisons between the first to 3rd-order all have p-values lower than 0.0002. This is in line with the computed weights for the matrices. These are high for the two directly preceding actions and then steeply decline. The prediction accuracy of both pick and place actions increases with model complexity in a similar manner. There is only a small difference between their accuracies in a direct comparison.

For the remaining analysis of the cross-validation, we pick the 3rd-order Markov chain as the representative for this model class. A closer inspection of the 3rd-order Markov chain in Figure 7.16b shows that only the accuracy for D-TASK2 is significantly different, but by only a small margin.



(a) Combined accuracy of different models for all datasets. The models from left to right are the 3rd-order Markov chain, the DNN, the fixed assembly rules, and random guessing from all goal-oriented actions.

(b) Accuracy of the **DNN** for different datasets.

Figure 7.17.: Accuracy of different models with **cross-validation**.

The comparison of Markov chains with the other classification algorithms is depicted in Figure 7.17a. The DNN performs better in terms of place actions (8 p.p.) but slightly worse for pick actions (by 5 p.p.). A closer inspection of the datasets in Figure 7.17b reveals that the DNN struggles to predict which object is picked when decomposing the structure. Despite having many more alternatives, the prediction of pick actions for D-TASK2 is 25 p.p. better.

A comparison between the accuracy of the 3rd-order Markov chain (Figure 7.16b) and the DNN (Figure 7.17b) shows that the DNN performs significantly better for place actions in D-TASK1 and D-DECOMP (by roughly 10 p.p.), whereas Markov chains excel when it comes to picking from the structure (by 13 p.p.). Differences are highly significant with p-values below 0.0002. All other differences between those two models are insignificant and small.

In terms of predicting place actions, both models can compete with the human reference from Figure 3.8. They perform worse only in pick predictions, with approximately 25 p.p. lower accuracy. We note, however, that the prediction task in Section 3.2 was different in terms of constructed structures and the number of alternatives to choose

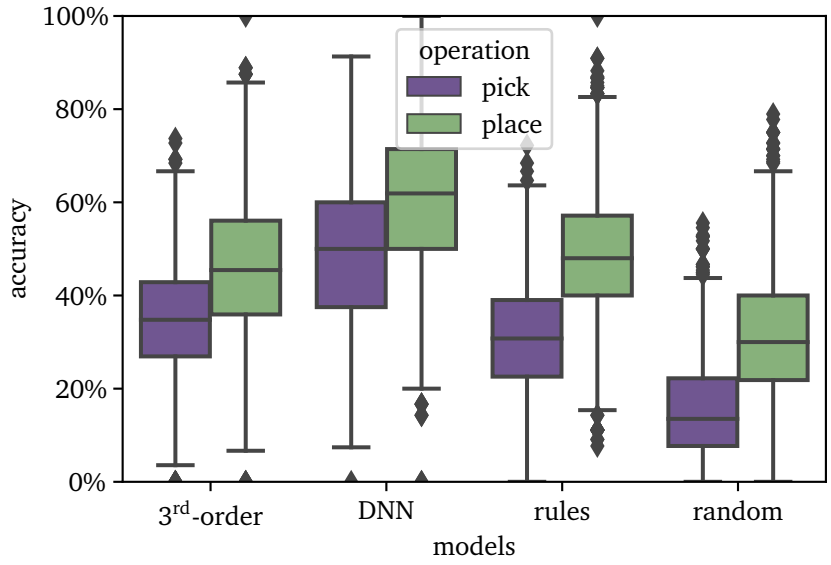


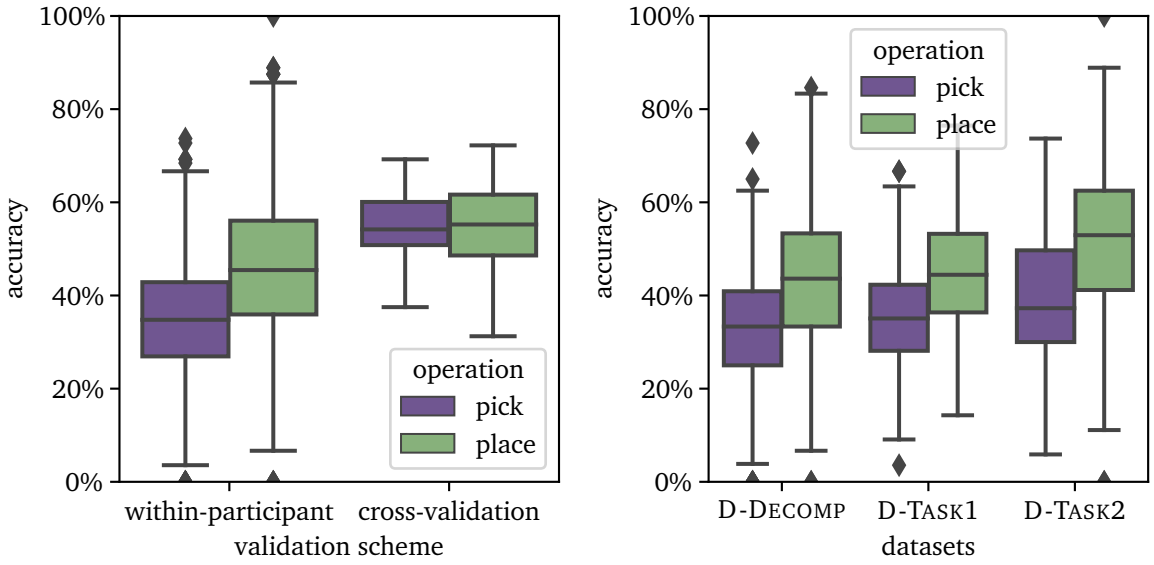
Figure 7.18.: Combined accuracy of different models for all datasets for **within-participant validation**.

from. Moreover, participants often didn't state a prediction if they were unsure; thus, the data in Figure 3.8 is biased towards those situations.

The fixed assembly rules fare significantly worse than both training-based approaches, but they are still better than random guessing. Fixed assembly rules achieve about 15 p.p. more accuracy than random guessing. So they capture some of the intrinsic behaviour patterns, but on a very coarse level. DNN achieves around 33 p.p. higher accuracy and the Markov chain 39 p.p. for pick and 24 p.p. for place actions.

To draw a conclusion from the cross-validation, all models have a decent performance and provide benefits in terms of predicting human behaviour. Both DNN and 3rd-order Markov chains have their strengths and weaknesses. So far, the models have been trained on a lot of data. Our goal, however, is to achieve good prediction capabilities with few training samples. Thus, we now compare how the models perform with fewer data in the within-participant validation. This validation scheme more closely resembles the study setup where the DNN was trained for each participant separately.

Figure 7.18 compares the models by accuracy when using within-participant validation instead of cross-validation. It is noticeable that variance is high for all models, including the random one. This stems from the few actions included in the validation set. Averaging fewer data points results more often in extreme outliers. Nevertheless, DNN and Markov chains can still learn relevant behavioural patterns setting them apart from random



(a) Combined accuracy for all datasets under different validation schemes.

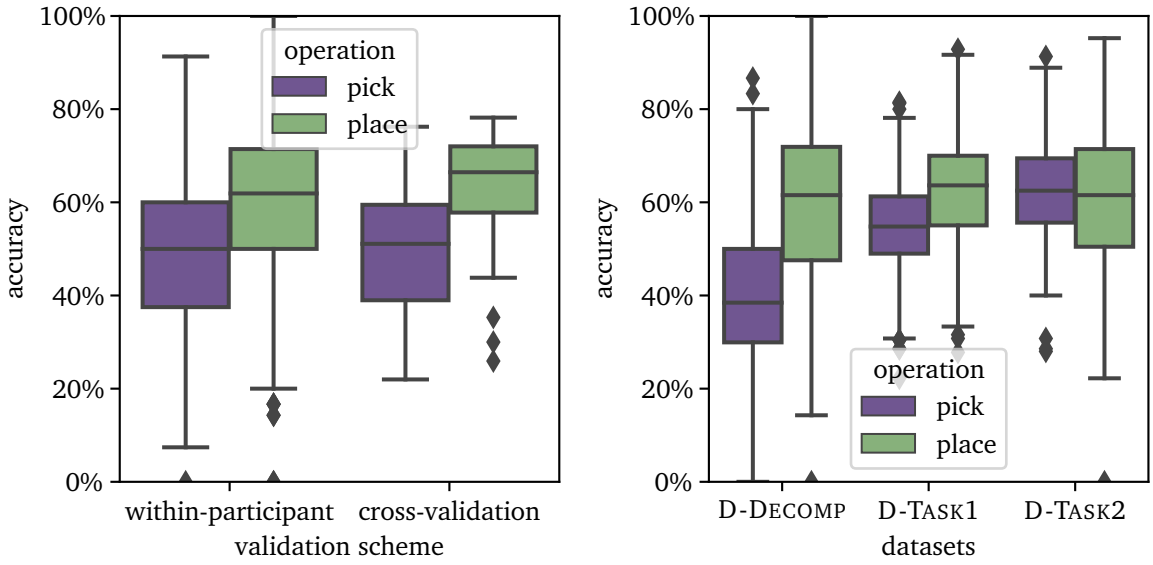
(b) Accuracy under the within-participant validation scheme for different datasets.

Figure 7.19.: Detailed comparisons for the **3rd-order Markov chain**.

guessing. But Markov chains suffer by large extents from the smaller training data. Their accuracy for place actions is just 3 p.p. better than fixed assembly rules, and the accuracy for place actions is even worse by 4 p.p. (both differences are still highly significant).

The direct comparison of the 3rd-order Markov chain under both validation schemes in Figure 7.19a shows that it loses 9 p.p. of accuracy for place actions and 20 p.p. for pick actions. Surprisingly, D-DECOMP, with its large number of alternatives, is not the main reason for the low prediction accuracy for pick actions, as shown in Figure 7.19b. Datasets D-TASK1 and D-TASK2 only perform up to 6 p.p. better. The differences are even smaller than in the cross-validation scheme (Figure 7.16b). That shows that the effect of many alternatives, where some rarely or never occur, has a lower impact on prediction capabilities than insufficient amounts of training data.

In contrast, the DNN demonstrates that behavioural patterns can still be learnt. Figure 7.20a shows that the DNN exhibits identical prediction accuracy for both validation schemes. That means it can very well cope with limited training data. An in-depth comparison of the datasets by putting Figure 7.17b and Figure 7.20b side-by-side shows that the DNN has the same strengths and weaknesses in both schemes. There are two factors that can explain the superiority of the DNN over Markov chains with limited training data: The first one is the feature space where decision-relevant information is directly



(a) Combined accuracy of the **DNN** for all datasets.

(b) Accuracy under the within-participant validation scheme for different datasets.

Figure 7.20.: Detailed comparisons for the **3rd-order Markov chain**.

encoded and does not need to be derived from action correlations. The second one is that the DNN is explicitly trained on the alternative actions as negative samples, whereas the Markov chain does not incorporate this information in the training process.

To investigate the training behaviour of the DNN further, we recreated the online training from the study. That means a training episode is run after every new action. The training data only consists of the first n runs of a trial—and not an arbitrary subset as in within-participant validation. Figure 7.21 shows the results when stopping the training after the first n runs and using the remaining ones as validation data. The stagnation of the accuracy is very apparent. The DNN no longer reaches the accuracy of the other two validation schemes. That is a strong indicator that the DNN is very sensitive to the initial data and might get trapped in local minima. It is therefore advisable to re-train the DNN from scratch with sufficient, recent data.

7.6. Conclusions and Discussion

This chapter presented a comprehensive evaluation of the complete software framework. As part of the study setup, three responsive robot behaviours were implemented (Section 7.2). All of them take the actions completed by the human into consideration. They

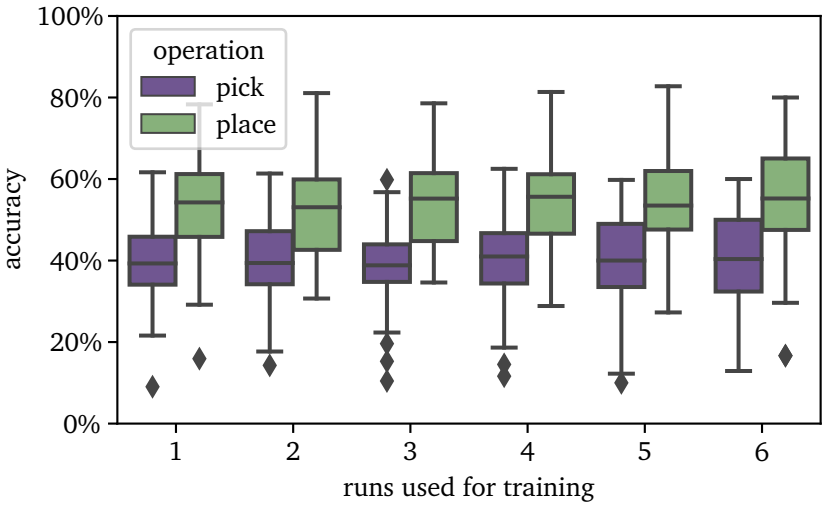


Figure 7.21.: Combined accuracy of the **DNN** for all datasets with online training.

regularly check the progress of the task to abort the current action if necessary. That means the robot can place an already picked object at a different location than previously planned, and it can put back an object that is no longer of any use. Three action selection strategies were implemented: one following a FIXED pattern, an ADAPTIVE one using the action prediction from Chapter 6, and an ADVERSARIAL one where the robot tries to mimic the user. The setup was evaluated with a user study. Study design was driven by ease of use, habituation, reproducibility, execution speed and flexibility. The participants had to repeatedly construct and dismantle structures of wooden toy blocks. This happened in a team with the robot, which followed one of the three action selection strategies.

The results from the fluency questionnaire show that the ADVERSARIAL strategy is perceived as less fluent. This indicates that the action prediction can predict relevant human behaviour patterns, e.g. picking objects close to the human or assembling the structure left-to-right. FIXED and ADAPTIVE are perceived as similarly fluent. As seen from the analysis of the DNN (Section 7.5), prediction results are good when properly trained. Three reasons can be identified why ADAPTIVE does not outperform FIXED: Firstly, the overall accuracy is suboptimal when trained incrementally, as during the study. Therefore, the DNN should be (re-)trained from scratch after several runs of data have been collected. Secondly, during the initial runs, the robot's behaviour changes and is sometimes perceived as random. Thirdly, taking the human's least preferred action is not necessarily the most supportive or least disturbing option. Other considerations,

such as workspace occupancy and overall completion strategy, play a role here, which were not explicitly modelled.

Main contributors for FIXED being perceived as more fluent than ADVERSARIAL are the positive teammate traits, improvement, working alliance, and subjective performance subscales. Robot relative contribution, trust, and goal perception do not have a significant effect. Correlations between objective productivity metrics and subjective fluency are often insignificant (Section 7.4.2). The same holds for the performance subscale of the NASA-TLX in comparison with the productivity metrics. Both indicate that it might be insufficient to measure productivity metrics to derive fluency or subjective performance metrics. However, fluency correlates with total load and, in particular, frustration of the NASA-TLX.

A dedicated analysis of the action prediction (Section 7.5) shows that the next action is correctly identified in $55\,\%$ of the cases. Especially in situations with limited training data, the DNN maintains its high performance and outperforms Markov chains or fixed assembly rules. To achieve that performance, the DNN needs a few runs of good-quality training data. Aggregating and cleaning the training data as done in Section 7.5 would therefore be necessary to achieve better results during system operation. This also reduces the problem of outliers and less productive solutions if initial runs do not exhibit characteristics of the desired cooperation. If the human, e.g., picks blocks from inconvenient positions to speed up the cooperation, the robot interprets this as a preference and acts accordingly in the next runs.

To cope with the problem of how to initially coordinate cooperative assembling, other researchers have proposed cross-training [55]. It is therefore worthwhile to further investigate how onboarding and the generation of initial training data can be efficiently combined. Initial training can then be used to adapt the robot's speed to the user's preference. Our study used a fixed robot speed for all participants. As investigated by Fratczak [203], some persons can fluently work alongside a high-speed robot while others pause until the robot is at a comfortable distance. The robot speed thus has an inter-participant influence on performance that can contribute to the high variance of subjective ratings. Subjective ratings also depended on whether the robot handled blocks far away from the user (e.g. the red blocks). Eight participants referred to this aspect in the open-form responses when rating the robot's behaviour. Therefore, further research is required to incorporate physical ergonomics into the robot's decision-making. The limited dexterity of the robot was a major constraint for the study design. Due to the point-to-point trajectories with several intermediate points where the robot stopped, it could not show its full potential. Therefore, the overall cooperation suffers from the human being

far more agile and completing more actions at the same time. Natural and fluent action execution is a long-standing problem in robotics research. It has recently seen a major boost by incorporating large neural networks and foundation models [192]. Future research in human-robot teaming can hopefully benefit from these advances. Despite this limitation, the study provides valuable insights into coordinating close human-robot teaming for assembly.

CHAPTER 8

Conclusions and Outlook

In this final chapter, we highlight the important contributions of this work, answer the research questions presented in Section 1.3, and point out future research. We start with a summary of this thesis.

8.1. Summary and Discussion

This thesis presented algorithmic and empirical contributions towards fluent human-robot teaming in the context of assembly work. The central hypothesis of this work is that subjective fluency benefits from behavioural adaptation based on action prediction. In particular, the goal was to set up a mutual adaptive human-robot teaming. With that in mind, task allocation should allow a maximum degree of flexibility. The robot should have the capability to perceive and predict the human's actions. In this context, answers are given to the research questions to summarise the insights from this thesis.

Q1 How effective is teamwork in the absence of speech? Which other channels of communication do humans use?

Chapter 3 described the re-creation of a human-human assembly study from the literature. Participants had to build four structures from wooden toy blocks as fast as

they could. Pre-assigned roles determined which blocks they were allowed to manipulate. There were not enough blocks to complete all the blocks simultaneously, but blocks from completed structures had to be reused. Both enforced coordination requirements. Contrasting the original study, participants were not allowed to talk during the assembly. Despite that restriction, participants were equally fast in completing the task (Section 3.1.2). A few participants expressed their discomfort with not being able to convey certain information verbally. Overall, participants felt comfortable executing the task and were content with their performance despite not being allowed to talk. Verbal communication thus seems not to be a necessary component for effective teamwork. An analysis of used gestures shows that participants most frequently 'point at part', 'point at location', and 'wave away'. Other communication channels, such as facial expressions or head gestures, were used far less frequently. These findings are in line with findings from another study that investigated gesture-based communication of assembly instructions. Both studies show the prevalence of these hand gestures for assembly tasks with a focus on pick-and-place actions.

Q2 How can the task state be efficiently updated under observation uncertainties in a flexible task model?

The literature review has shown that most approaches for action detection rely on motion information and are targeted towards daily actions. Action detection based on subcomponents of an assembly is instead a novelty with its own advantages and disadvantages. The advantages are the recovery from missed observations and the robustness against variance. The disadvantages are the necessity of a fine-grained object detection mechanism and the limitation to actions with clearly observable, persistent effects. This thesis presented a detection pipeline to obtain occupancy and occlusion information about all objects of the workspace in Section 5.2. Section 5.3 introduced coloured Petri nets with extended semantics to model the task state and all possible executions. Places represent physical locations and hands/grippers, whereas tokens represent the objects. The task model offers full flexibility in action allocation, ordering, and parallel execution, while respecting dependencies between actions. Sensory input is integrated into the model in terms of emissions originating from places. In contrast to classical Petri net settings, where transitions are directly observable, these emissions create new challenges. From the emissions, the fired transitions and new marking need to be deduced. Section 5.4.2 introduces a three-step algorithm to filter relevant transitions and then run a Monte-Carlo sampling. The result is a probability distribution over markings. If observations are contradicting, no updates are made until a valid marking can be recovered. This

conservative approach minimises the incorrect detection rate below $5\,\%$ on average (Section 5.7). Still, the overall pipeline has a low latency of $250\,\mathrm{ms}$. The explicit detection of emptiness makes it faster than DECAY-based approaches that require an object to fade out for several seconds. A drawback of the update is that there is no time limit. In case the emissions deviate more and more from the current marking, the update step takes longer, up to several seconds. Further constraints are thus required to adhere to real-time limits.

Q3 To what extent can human observers with limited sensory input predict action sequences of other humans working on an assembly task?

The perception capabilities of robots are still limited and different from human perception. The key idea for investigating this aspect is to reduce the presented information to the key events a robot registers on one side (Section 3.2.1), but on the other side, to provide a representation easily comprehensible by humans. Both are realised by videos of the assembly objects, but real objects are replaced by virtual ones. The presence and absence of virtual objects are determined by a hypothetical perception pipeline. For instance, objects that are partly visible or being manipulated are not shown. Researcher colleagues were then asked to watch the videos and identify intentions with open-form responses. For $60\,\%$ to $80\,\%$ of the events shown, they stated a prediction, which was correct in around $60\,\%$ of cases (Section 3.2.2). This is around 20 p.p. better than guessing. Prediction accuracy is not influenced by the number of alternatives that exist. When hand positions are also visible, prediction accuracy increases by around 10 p.p. To conclude, despite limited sensory input, humans are well capable of forming shared mental models and predicting their assembly behaviour.

Q4 To what extent can action sequences of humans be algorithmically predicted from previous behaviour?

The literature review has shown that everyday activities are the most common application scenarios for action prediction (Section 6.1). Most approaches utilise CNNs that take video streams as input and output action labels. The large amount of training data makes them unsuitable for the domain of industrial assembly tasks. The domain of event prediction deals with similarly structured problems. Approaches for event prediction can cope with smaller training datasets by utilising neural networks if the number of categories is low enough. The classical approach for event prediction is to input a fixed-length vector of class labels from previous actions and to output a probability distribution over

all labels. That approach has two disadvantages: (i) common properties among actions are lost if only a class label is provided, and (ii) inactive actions influence the training process. To tackle the first disadvantage, we take inspiration from natural language processing and encode actions as feature vectors. The encoding is fixed and includes position, colour, volume, and neighbourhood information (Section 6.3). Since only a small subset of actions can be executed in each task state, the problem of action prediction is broken down into the following: given the previous four actions and a candidate action, output its likelihood (Section 6.4). Likelihoods for all candidates are compared with each other to rank the actions and form a probability distribution. That formulation allows for handling variably sized sets of candidates for the next actions. The novel formalisation of the problem enables the deployment of a small, deep neural network to learn the input-output relationship. The DNN has the desired trait that it can be trained during interaction with the robot, and thus adapt to the human (Section 6.5). The accuracy of the DNN is evaluated on the human-robot teaming study from Chapter 7. With a mean accuracy between 50 p.p. and 60 p.p., the DNN is in the range of human prediction capabilities (Section 7.5). The comparison with Markov chains shows that the DNN is 15 p.p. more accurate. Using fixed rules for predicting assembly sequences (Section 6.2) cannot compete with trainable models. The evaluation shows that SMMs can be efficiently learnt with small-sized models without pre-training. This opens opportunities for adaptive robot behaviour based on human preferences.

Q5 How does action prediction impact cognitive ergonomics in an industrial-like human-robot teaming scenario? What is the specific effect on fluency?

Section 7.1 presents a prototype implementation of a human-robot teaming setup with mutual adaptation. The reference task is the construction of structures from wooden toy blocks. The task allows easy habituation by humans and the execution of all actions by both agents. The robot uses the tracking system from Chapter 5 and the prediction of human actions from Chapter 6. Both agents dynamically choose their next actions after completing one, enabling high autonomy and neglect tolerance for both of them. Three strategies are implemented for the decision-making of the robot (Section 7.3). All strategies take actions completed by the human into account and flexibly choose one of the active actions. The FIXED strategy uses a fixed order of preference to choose the next action. The ADVERSARIAL strategy tries to choose the same action the user would perform next. Finally, the ADAPTIVE strategy chooses the action the human executes least likely. The evaluation shows that ADVERSARIAL was perceived as least fluent, showcasing the effect when human and robot action selections are in conflict. The most influential factors

for that rating are positive teammate traits, working alliance, and subjective performance. Despite the fact that action prediction can achieve high accuracy (Section 7.5), the ADAPTIVE strategy is not perceived as significantly more fluent than the fixed, though a positive trend can be observed (Section 7.4.1). The reasons are the high variance in individual ratings, the incremental training of the DNN (Figure 7.21), and the robot's decision-making that could be improved in terms of choosing truly supportive actions. Overall, this thesis indicates that intention prediction can positively impact some aspects of cognitive ergonomics. It shows that misalignment of coordination, e.g. stemming from inappropriate schedules, has a severe negative impact on cognitive ergonomics. The central working hypothesis is thus only partly confirmed: misaligned SMMs lead to decreased fluency, but a significant improvement in fluency from action prediction could not be observed.

Q6 How are objective and subjective metrics with respect to productivity and fluency related?

Finally, the results of the human-robot teaming study are used to gain insights into the correlations between metrics of fluency, mental workload, subjective, and objective performance (Section 7.4.2). As a first step, task durations are normalised to remove between-subject variance stemming from different base working speeds. As a first insight, task durations show significant differences between robot behaviours but do not correlate with fluency. Other objective performance metrics, such as concurrent activity, human idle, and robot idle time, do not correlate with subjective fluency either. The only exception is the subscale of robot contribution that correlates with human idle time. In line with previous studies, the conclusion is that fluency cannot simply be derived from objective performance metrics. In terms of cognitive ergonomics, fluency correlates with total load from the NASA-TLX. The most influential subscales for the correlation are frustration and trust. To conclude, perceived fluency is an orthogonal metric compared to objective performance, but its effects can be captured by the established NASA-TLX. If both metrics are used, fluency can provide more detailed insights into what contributes to high total load, and fluency can statistically differentiate finer nuances between experimental conditions.

8.2. Future Work

This work can be extended on multiple levels: improvements of individual components, further research directions, and research practices.

Improvements of Components

The task state tracking can be backed by a more capable and robust object detection framework. Bringing the camera closer to the workspace or installing a second one with only the purpose of observing the workpiece are two promising directions. The current task state tracking algorithm achieves a precision of more than $95\,\%$. However, that still causes several incorrectly detected workspace states within one trial of the user study, as the data preprocessing in Section 7.5 showed. The main cause is empty places incorrectly identified as occupied. Improving the object detection component can have major benefits for precision and versatility. In terms of versatility, the robust detection of object states, e.g. placed, screwed in, and sealed screws, enables the tracking of tasks that reach beyond pick-and-place. Another point is the extension towards workstations integrated into an assembly line where new parts are successively delivered. The capabilities of the model would need to be extended towards controllable transitions that can generate an infinite number of tokens. When it comes to extending the task state tracking towards assembly lines, strict real-time constraints become important. Though the current implementation requires only a few milliseconds on average, there are outliers that require several seconds. Replacing the fixed number of samples in the Monte-Carlo sampling with a time limit is a first step in this direction.

Action prediction shows good results for the isolated evaluation (Section 7.5). However, the analysis revealed that more effort must be invested to achieve robust incremental learning. The current approach suffers from the gradient descent getting stuck in local minima when the initial training data is too small. In future studies, more data from one human should be collected before starting the training. As a consequence, an initial behaviour must be selected. Like in the study, the robot can follow a fixed pattern, a pre-trained DNN from general human behaviour or other approaches from cross-training can be explored to improve onboarding of the human-robot team. Incremental long-term learning is another aspect that is not addressed in this thesis. The current approach does not take into consideration that when the system runs for days, the accumulated data exceeds the physical memory. For an actual deployment, the approach thus requires further adjustments. From the scientific perspective, an in-depth analysis of the learnt behaviour can show whether the design goals are met. Explainable AI approaches can show whether the DNN utilises the neighbourhood information to encode order patterns. Insights in this part are a necessary step towards expanding the action prediction for assembly steps that go beyond pick-and-place.

Regarding the human-robot teaming study, several design decisions prohibit a direct

comparison with workstation setups deployed in industry. Firstly, even the FIXED behaviour detects completed actions and skips them. Current industrial deployments use fixed schedules that are planned offline instead. Secondly, the selected pick-and-place tasks require low precision and can therefore be executed very fast by humans. That further reduces the relative robot contribution. Thirdly, the productivity and job quality of workstations should stay high over the course of a whole working day. A user study with a total length of 1 h only provides limited insight. To judge the benefits of mutual adaptation in comparison with current setups, a study that is closer to industrial setups with regard to those three aspects would be necessary.

Research Directions

The selection of tasks and actions in this thesis is mainly driven by the technical limitations of the available robot system and the design goal of flexibility. Task characteristics and assembly steps of real industrial use cases only had an indirect influence. Systematically identifying processes where high flexibility is desired, the required perception capabilities, and assembly steps is the next step towards a research prototype deployable in an industrial scenario. The identified process steps then inform us in which directions the task state tracking and features for action prediction need to be extended. Aspects of safety certification, workspace design, and privacy are out of scope for this thesis, but would need to be considered when deploying this prototype.

The task state tracking offers limited capabilities for error detection. If there is no sequence of transitions to reach a marking compatible with the current emission, then no update is made (Section 5.4.2). For the user, it remains unclear whether this results from processing time or incorrect task execution when the remaining task steps are not updated. Evolving the task model towards detecting individual sources of errors is therefore a promising direction. It allows for the alleviation of the assumption that humans carry out the task without errors. That is neither perfectly valid for lab experiments (Section 3.1.2) nor industrial scenarios [207].

Section 7.4 identifies a dictionary of gestures to achieve joint attention for assembly in human teams. In the course of this work, the integration of gestures has been disregarded due to its complexity and technical limitations. However, newer devices and software frameworks, e.g. the Meta Quest 3 [81], show a promising direction towards hand tracking and gesture recognition. With gesture recognition readily available, gestural communication from human to robot can be explored. Unlike keyboard or mouse input, gesture-based input integrates seamlessly into the workflow without a tool switch.

Gestures can then be used to reassign another action to the robot if the planned one does not meet the user's preferences. Positive impacts can be expected for fluency, as some participants stated that the robot did not follow their expectations in the ADAPTIVE strategy. Instructions are then an accompanying feature besides the proactive action execution. The user study has moreover shown that selecting an appropriate action by the robot is more complex than selecting the least disturbing one. In addition to direct instruction, it might be worthwhile to investigate further approaches and aspects for the decision-making of the robot.

Research Practices

Robotics research in the past decades suffered from bad publication practices. Many publications only include plots of the evaluation and a coarse description of the software framework. Reimplementing the frameworks is basically impossible—just like comparing different approaches, as evaluations used completely different evaluation strategies, benchmark tasks, and metrics. Driven by these obstacles, the term *Reproducible Robotics Research* has gained momentum [208]. Since then, workshops, an article series for publishing reproducible research [209], and frameworks for rapid prototyping [210, 211] have been established. Reproducibility is achieved by four pillars. The first pillar is the precise descriptions of the algorithms, parameters, study setups, and results in a written work. This grants a basic understanding of the core parts of the framework. The second pillar is publishing the source code or executable binaries. The best solution would be to publish containerised software that runs on most desktop computers. The third pillar is the publication of data collected during the experiments. These include the raw data of the studies and the processing used to generate the plots. The fourth pillar is the publication of hardware descriptors of all components. This thesis contributes to the first pillar. The source code of the overall framework [212], the data from the experiments, and Jupyter notebooks to generate the plots [213] are published alongside. Attempts have been made towards containerising the experiments in Section 5.7. However, the large number of dependencies and system libraries required for the container resulted in regular updates for the build script being necessary. It therefore remains up to the future to make the task model update procedure and action prediction fully reproducible. The full reproducibility of the whole setup has already become infeasible since the HoloLens 2 and Kinect 2 have been deprecated. Making human-robot teaming approaches reproducible is thus an open problem. The complex hardware and software setups, as well as the fact that humans are individual—and as such, a different sample of humans might behave differently—pose significant challenges.

APPENDIX A

Human-Human Coordination Study

A.1. Study Procedure

1. Two participants are brought into the room
2. The participants receive the information sheet for the study
3. The participants fill out and sign the consent form
4. The interviewer explains the task to the participants and assigns the roles
5. The interviewer puts the trackers on the participants
6. The participants have time to familiarise themselves with the gloves and the task (up to 15 min)
7. The interviewer explains the rules for the implementation
    - 3 phases, in each of which all 4 structures must be set up
    - any order
    - parallel work allowed
    - Ok from the interviewer for each completed structure
    - Structure does not have to remain intact after completion
    - grasp max. 1 block per hand
    - No talking allowed

- 5 min break for dismantling the structures, strategic counselling and training

8. The interviewer starts Kinect, tracking and camcorder
9. The participants work on the task
10. The interviewer instructs participant 1 to wait outside the door
11. The interviewer asks participant 2
12. The interviewer questions participant 1
13. The interviewer explains the purpose of the study and offers a tour of the Botlab
14. The interviewer copies the data to the external hard drive

The following two pages contain the instructions for the participants.

Informationen zum Experiment

Generelle Informationen

In diesem Experiment geht es darum zu evaluieren, wie zwei Menschen zusammenarbeiten. Sie werden hierfür im Team die nachfolgend abgebildeten vier Gebilde aufbauen. Dabei wird die Bewegung Ihrer Hände per motion-tracking erfasst. Zusätzlich werden Sie von zwei Kameras gefilmt.

Phasen

1. Informationen zum Ablauf, Einverständniserklärung und Anbringen der Tacking-Marker
2. Eingewöhnungsphase: Machen Sie sich mit der Aufgabe vertraut. Sie dürfen Teile der Gebilde probeweise aufbauen und miteinander sprechen. Am Ende der Phase müssen alle Bauklötze wieder im ursprünglichen Bereich liegen. Sie dürfen dabei diskutieren, sich Strategien überlegen und die Klötze neu anordnen.
3. Durchführung: Sie erhalten weitere Instruktionen vom Versuchsleiter
4. Befragung der Teilnehmer

Hinweise

- Achten Sie darauf, die an Ihrer Hand befestigten Marker nicht zu verdecken
- Versuchen Sie die Hände möglichst im Bereich der grünen Matte zu belassen (andernfalls kann das Tracking der Hände stellenweise Aussetzen)
- Beugen Sie sich nicht zu weit vor, sondern sitzen Sie aufrecht (andernfalls verdecken Sie mit Ihrem Kopf die Gebilde)

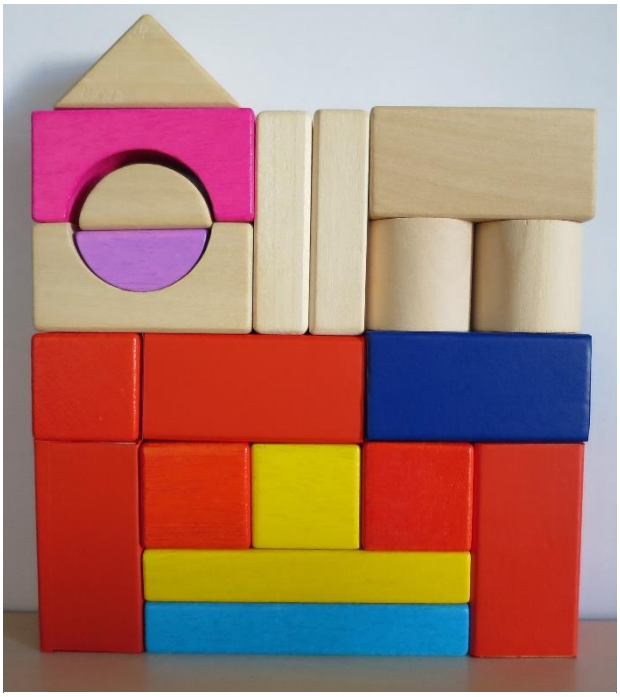
Aufgabe

Nachfolgend sind vier Gebilde abgebildet, die sie gemeinsam aufbauen sollen. Das genaue Vorgehen ist Ihnen freigestellt, jedoch müssen folgende Regeln eingehalten werden:

- Ein Gebilde darf **zerstört** und die Bauklötze wiederverwendet werden, **nachdem** sie **vollständig** aufgebaut wurde
- Jedem ist eine **Rolle** zugewiesen. Entsprechend der Rolle darf nur einen Typ von Bauklötzen (bunt vs. Holzmaserung) bewegt werden.
- Sollte ein Gebilde (teilweise) einstürzen, darf diese soweit nötig abgebaut bzw. zurechtgerückt werden. Die Rollenverteilung ist beim Abbau zu berücksichtigen.

Gebilde #1

Gebilde #2



Gebilde #3

Gebilde #4

Experiment Information

General Information

In this experiment, we want to evaluate how two humans cooperate. Therefore, you will build the following four structures as a team. A motion tracking system will capture the movement of your hands. Additionally, two cameras will film you.

Phases

1. Information regarding the procedure, consent form, and fixing the tracking markers
2. Familiarization phase: Familiarize yourself with the task. You are allowed to build parts of the structures for training and to talk to each other. At the end of the phase, all building blocks must be in the original area. You are allowed to discuss, strategize and organize the building blocks
3. Execution: You will receive additional information
4. Questioning the participants

Notes

- Please take care of not occluding the markers attached to your hand.
- Try to stay with your hands within the area of the green mat.
- Please do not bend too far forward, but try to sit straight (otherwise, your head may occlude the structures).

Task

In the following four structures are presented that you shall build up cooperatively. The precise procedure is up to you. However, you have to observe the following rules:

- A structure may be **destroyed** to reuse the building blocks, **once** it is completely **finished**.
- Each of you has a **role** assigned. Accordingly, you are only allowed to manipulate one type of building block (colored vs. tan)
- In case a building (partially) collapses, you are allowed to dismantle, or adjust, it as far as necessary. But you have to stick to your roles when dismantling

A.2. Evaluation

Operation	Comparison	Wilcoxon rank-sum	Hodges-Lehmann-Sen shift estimator
pick	random goal-oriented vs. blocks	4.239594e-11	0.5 within [0.3333, 0.6667]
pick	random goal-oriented vs. blocks & hands	9.485738e-15	0.4598 within [0.4524, 0.5]
pick	blocks vs. blocks & hands	6.640109e-01	-0.06667 within [-0.2069, 0.1264]
place	random goal-oriented vs. blocks	1.628148e-06	0.04167 within [0.04167, 0.2083]
place	random goal-oriented vs. blocks & hands	7.585532e-04	0.2 within [0.1786, 0.2]
place	blocks vs. blocks & hands	1.677010e-02	0.1583 within [-0.07143, 0.1583]

APPENDIX B

Human-Robot Fluency Study

B.1. Hardware Description

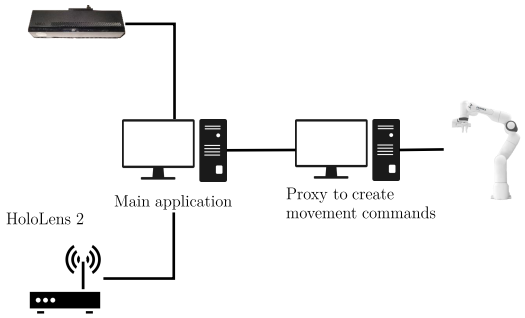


Figure B.1.: Connection of hardware components. The Kinect 2 (top left) is connected by USB. The HoloLens 2 communicates with the main computer via a wireless LAN connection. Both sides use Protobuf [214] to handle bidirectional event processing and remote procedure call on the basis of a custom, language-independent protocol. All other connections are based on Ethernet.

List of hardware components:

- Franka Emika Robot[1] with firmware version 3.2.0 and Schmalz CobotPump[2] and a seduction cup of $2\,\mathrm{cm}$ in diameter [3]

[1] https://web.archive.org/web/20201101041759/https://franka.de/ (date accessed: 2025-01-29)
[2] Model: ECPPi 12 48V-DC M12-5, new models: https://web.archive.org/web/20201101041759/https://franka.de/ (date accessed: 2025-01-29)
[3] https://www.schmalz.com/en/vacuum-technology-for-automation/vacuum-

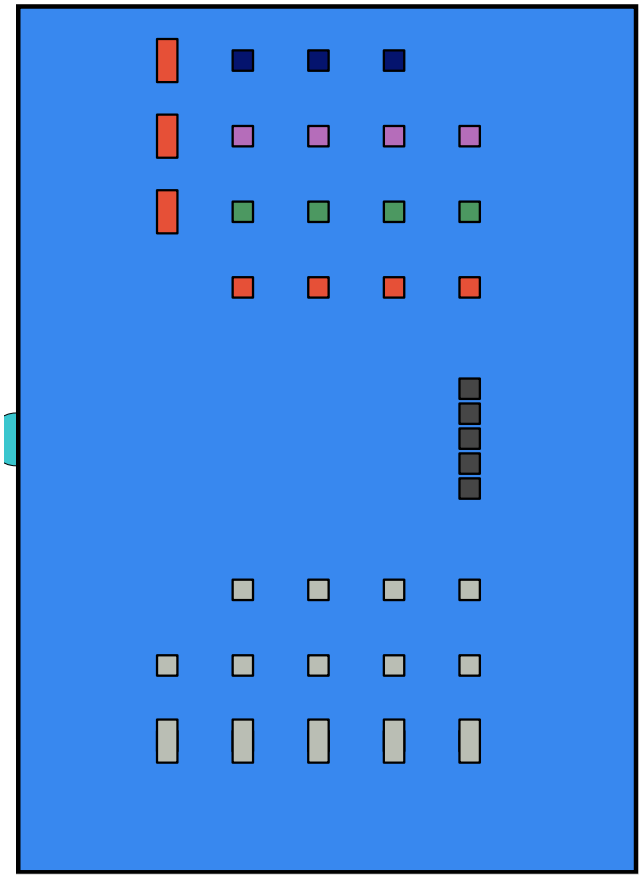


Figure B.2.: Mat used for the study in a scale of 1:10. The mat is $82\,\mathrm{cm} \times 115\,\mathrm{cm}$, excluding the cut-out part for the robot. The centres of the cubes of the resource pool are $10\,\mathrm{cm}$ apart. The construction area for the structure is $60\,\mathrm{cm}$ from the robot base.

- Kinect 2[4]
- Desktop Computer with Intel i7-7700K processor, $32\,\mathrm{GB}$ of DDR4 memory, and Windows 10 Build 19045 for the main application [212]
- Desktop Computer with Intel i7-6700K processor, $16\,\mathrm{GB}$ of DDR4 memory, and Windows 10 Build 19045 for the proxy server
- HoloLens 2 [80]
- TP-link wireless router (TL-WR841N) for connection with HoloLens 2
- Wooden toy blocks with edge lengths of $3\,\mathrm{cm}$, with[5] and without colouring[6]
- Mat for layout of blocks: Figure B.2

Figure 7.1 depicts the spatial arrangement of all relevant entities. Figure B.1 depicts the connection of hardware components. The reference coordinate system is the one of the robot, rooted in its base. All other coordinate systems are registered with respect to the reference coordinate system.

To register the Kinect 2, *RANSAC* [215] is run on the point cloud to find a plane S for the surface of the workstation. Next, the point cloud is transformed such that the S aligns with the x-y plane of the reference coordinate system. In the next step step, the origin and z-rotation of S need to be calculated. Two distinguishable reference objects are placed on the mat. The mat is designed to perfectly attach to the robot base and therefore align with the reference coordinate system. The position of the reference objects within the surface-aligned point cloud is calculated using the pipeline described in Section 5.2. This provides an initial estimate of the coordinate transformation between the local coordinate system of the Kinect 2 and the reference coordinate system. The transformation is manually fine-tuned by visually comparing the points of the point cloud with the position of the objects in the reference coordinate system.

components/vacuum-suction-cups/suction-cups-for-the-packaging-industry/bellows-suction-cups-spb1-1-5-folds-304469/10.01.06.02452/ (date accessed: 2025-01-29)

[4] https://web.archive.org/web/20180313031651/https://blogs.msdn.microsoft.com/kinectforwindows/2014/07/15/the-kinect-for-windows-v2-sensor-and-free-sdk-2-0-public-preview-are-here/ (date accessed: 2025-01-29)

[5] https://www.amazon.de/dp/B082YN8F54/ref=sspa_dk_detail_0 (date accessed: 2025-01-29)

[6] https://www.amazon.de/dp/B000O3AKTO/ref=pe_3044161_185740101_TE_item (date accessed: 2025-01-29)

The registration procedure, data transfer, and creation of the augmented reality for the HoloLens 2 were implemented in a student's project [216]. For the registration, the user moves around the work station to generate a sequence of point clouds. These are sent to the computer running the main application and merged into a single point cloud. Next, *Iterative Closest Point* [217] finds the transformation between the merged point cloud and the one obtained from the Kinect 2, for which the transformation to the reference coordinate system is already known. The achievable accuracy is within a few millimetres. The location of the robot's origin is then transferred to the HoloLens 2 to create a spatial anchor [218]. The HoloLens 2 detects the spatial anchor when powering on. Object holograms can thus be placed relative to the spatial anchor and therefore relative to the reference coordinate system. Absolute position accuracy of the rendered objects $60\,\mathrm{cm}$ away from the robot – the construction place of the structure – is around $1\,\mathrm{cm}$.

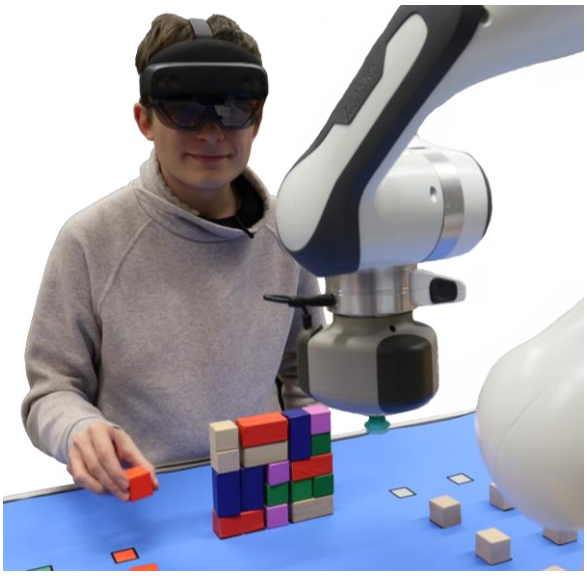
B.2. Material

The following pages of this thesis contain the material provided for the participants of the human-robot study described in Section 7.3:

- Instruction sheet in English and German: pages 165 to 168
- Consent form in German and English: pages 169 to 170
- NASA-TLX: [219]
- Fluency questionnaire in English and German after each trial (English version from [21]): pages 171 to 174
- Questionnaire in English and German at the end: pages 175 to 182

User Study: Human-Robot Cooperation

Background



You see a table with building blocks and a robot in front of you. In the course of the study, you will repeatedly build and dismantle one structures together with the robot. This is to simulate a factory scenario with several production cycles. By dismantling the structures, it is possible to simulate several production cycles without having to constantly supply new material.

Procedure

First, you put on the HoloLens and familiarise yourself with its operation. The experimenter will support you in this. If problems occur (e.g. blocks are displayed several times, are missing, or are still not recognised after several seconds), please inform the trainer. If necessary, the application must be restarted on the HoloLens.

Then familiarise yourself with the display and reaction of the system. The HoloLens uses holographic blocks to show you which structure you need to build. Position the HoloLens on your head such that it feels comfortable and you accurately perceive the colours (even a minor adjustment of the display position dramatically affects colour perception). The position estimation of the HoloLens is not completely accurate, i.e. the holographic blocks may be slightly displaced. In this case, stick to the markings on the mat. This displacement does not affect how the robot perceives the environment.

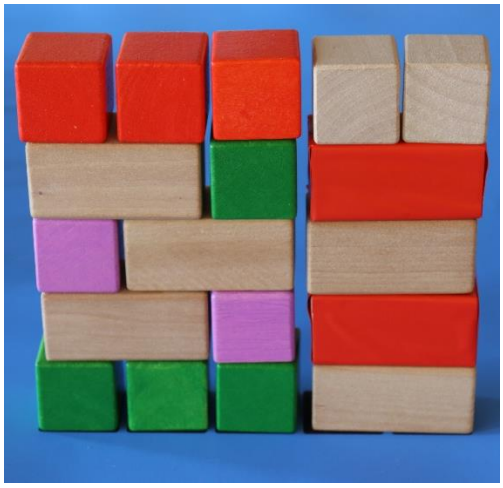
You will build and dismantle one structure on a trial basis. Placed and recognised blocks disappear. However, this does not apply if you place a wrong block. Return the block to the place where it was picked up before continuing. When a structure is completely built, the holographic blocks are displayed on the marked resource places. Only then has the system recognised the structure as finished and you may start dismantling it. The recognition is done by means of the depth camera mounted on the ceiling. So if placed blocks are not recognised, take half a step back and move your hands towards your body.

The study consists of a total of **3 runs of 25 min each**. In each run, you will rapidly build up and dismantle several structures. Continue until the experimenter ends the run. At the end of each run, a questionnaire is to be filled out with which you evaluate the cooperation with the robot.

You can stop the study at any time without giving a reason.

As an orientation, the two possible structures are shown here. However, you do not have to remember their composition.

Structure 1



Structure 2

Please obey to the following rules when building up and dismantling the structures:

- **Blocks must not be rotated under any circumstances**
- **Only place blocks on the marked places.**
- **Do not deliberately knock over or move the blocks.**
- Do not hold more than one block in your hands at a time
- Both hands may be used
- However, try to avoid switching a block from one hand to the other
- Both hands should never be below the table nor crossed (so that the HoloLens recognizes them correctly)
- If a structure collapses **during construction**, restore it to the state it was in before the collapse and continue.
- If a structure collapses **during dismantling**, continue dismantling.
- Long blocks may not be replaced by two cubes.

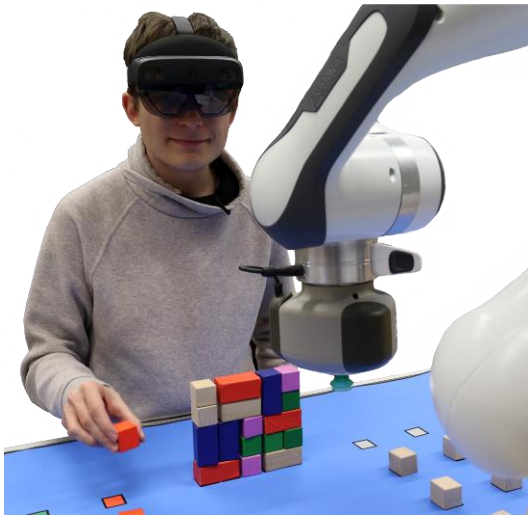
Cooperation with the robot

In most (but not all runs) you will build up the structures with the robot. **The experimenter will stop the robot immediately if there is a risk of injury or collision**. The place where the robot picks up or puts down the next block will be outlined in bright red. This is to help identify the robot's plan. The robot will always make its decisions based on the progress shown to you on the HoloLens. Changes you make while the robot gripper is close to or above the target can therefore not be taken into account and can lead to errors. It can also happen that the robot places blocks in a very offset position. In this case, please correct the position.

Each run starts at 0, i.e. without the robot having knowledge about previous runs. You can also make a conscious decision to have the robot place a particular block instead of placing it yourself. Note, however, that the robot can only pick up and place blocks where its suction pad does not collide with other blocks.

Nutzerstudie: Mensch-Roboter-Zusammenarbeit

Hintergrund



Sie sehen vor sich einen Tisch mit Bauklötzen und einem Roboter. Im Zuge der Studie werden sie zusammen mit dem Roboter wiederholt ein Gebilde errichten und abbauen. Dies soll ein Fabrikszenario simulieren mit mehreren Produktionszyklen simulieren. Durch das Abbauen der Gebilde wird ermöglicht, mehrere Produktionszyklen zu simulieren ohne permanent neues Material nachreichen zu müssen.

Ablauf

Als erstes setzen Sie die HoloLens auf und machen sich mit der Bedienung vertraut. Der Versuchsleiter wird Sie dabei unterstützen. Sollte Probleme auftreten (z. B. Blöcke werden mehrfach angezeigt, fehlen, oder werden nach mehreren Sekunden immer noch nicht erkannt), weisen sie bitte den Versuchsleiter darauf hin. Ggf. muss die Anwendung auf der HoloLens neu gestartet werden.

Anschließend machen Sie sich mit der Darstellung und Reaktion des Systems vertraut. Die HoloLens zeigt Ihnen mittels holographischer Blöcke, welches Gebilde sie errichten müssen. Richten Sie sich die HoloLens so ein, dass sie komfortabel sitzt und Farben korrekt wiedergegeben werden (selbst minimales Verschieben der Anzeige beeinflusst die Farbwahrnehmung dramatisch). Die Positionsbestimmung der HoloLens ist nicht ganz exakt, d.h. die holographischen Blöcke können leicht verschoben sein. Richten Sie sich in dem Fall nach den Markierungen auf der Matte. Die Verschiebung beeinflusst nicht, wie der Roboter die Umgebung wahrnimmt.

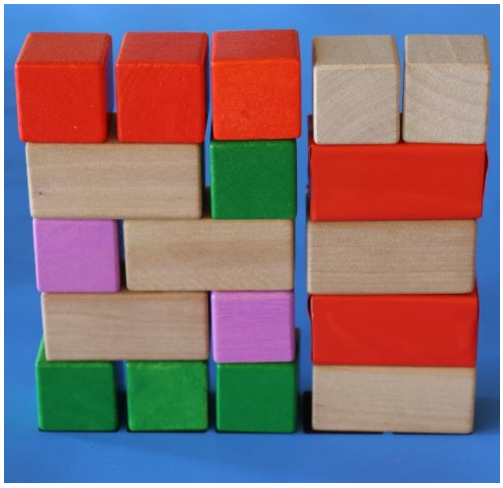
Sie werden probeweise ein Gebilde auf- und abbauen. Platzierte und erkannte Blöcke verschwinden. Dies gilt jedoch nicht, wenn sie einen falschen Block platzieren. Legen sie den Block an den Aufgreifort zurück, bevor sie fortfahren. Ist ein Gebilde vollständig errichtet, werden die holographischen Blöcke auf den markierten Ressourcenplätzen angezeigt. Erst dann hat das System das Gebilde als fertig erkannt und sie dürfen mit dem Abbau beginnen. Die Erkennung erfolgt mittels der an der Decke angebrachten Tiefenbildkamera. Wenn also platzierte Blöcke nicht erkannt werden, gehen Sie einen halben Schritt zurück und bewegen Sie die Hände zum Körper.

Die Studie besteht insgesamt aus **3 Durchgängen zu je 25 min**. In jedem Durchgang werden sie zügig mehrere Gebilde auf- und abbauen. Fahren Sie solange fort, bis der Versuchsleiter den Durchlauf beendet. Am Ende jedes Durchgangs ist ein Fragebogen auszufüllen, mit dem Sie die Zusammenarbeit mit dem Roboter bewerten.

Sie können die Studie jederzeit ohne Angabe von Gründen abbrechen.

Als Orientierung sind hier die zwei möglichen Gebilde dargestellt, von denen Sie eines bauen werden. Sie müssen sich die Zusammensetzung aber nicht merken.

Gebilde 1



Gebilde 2

Beachten Sie beim Auf- und Abbauen bitte folgende Regeln:

- **Blöcke dürfen keinesfalls gedreht werden**
- **Blöcke immer nur an den markierten Orten ablegen**
- **Gebilde nicht absichtlich umstoßen oder verschieben**
- **Maximal einen Block gleichzeitig in der Hand halten**
- Es dürfen beide Hände verwendet werden
- Vermeiden Sie aber einen Block von einer in die andere Hand zu wechseln
- Beide Hände sollten sich niemals unter dem Tisch befinden und niemals verschränkt sein (damit die HoloLens sie korrekt erkennt)
- Sollte ein Gebilde **während des Aufbauens** einstürzen, stellen sie den Zustand vor dem Einsturz wieder her und fahren sie fort.
- Sollte ein Gebilde **während des Abbauens** einstürzen, fahren sie mit dem Abbau fort.
- Lange Blöcke dürfen nicht durch zwei Würfel ersetzt werden

Zusammenarbeit mit dem Roboter

In den meisten (aber nicht allen Durchläufen) werden Sie die Gebilde mit dem Roboter errichten. **Der Versuchsleiter wird den Roboter sofort anhalten, falls die Gefahr einer Verletzung oder Kollision besteht.** Der Ort, an dem der Roboter den nächsten Block aufgreift bzw. ablegt, wird knallrot umrandet. Dies soll dabei helfen, den Plan des Roboters zu erkennen. Der Roboter geht immer von dem Fortschritt aus, der Ihnen auf der HoloLens dargestellt wird. Änderungen, die sie vornehmen, während sich der Robotergreifer nahe oder über dem Ziel befindet können folglich nicht berücksichtigt werden und können zu Fehlern führen. Ebenfalls kann es passieren, dass der Roboter Blöcke **stark** versetzt ablegt. In dem Fall bitte geraderücken.

Jeder Durchlauf beginnt bei 0, d.h. ohne dass der Roboter Wissen aus vorherigen Durchläufen mitnimmt. Sie können sich auch bewusst dafür entscheiden, einen bestimmten Baustein nicht selbst sondern vom Roboter platzieren zu lassen. Beachten Sie allerdings, dass der Roboter Blöcke nur dort aufgreifen und ablegen kann, wo sein Sauggreifer nicht mit anderen Blöcken kollidiert.

UNIVERSITÄT BAYREUTH

Lehrstuhl für Angewandte Informatik III
(Robotik und Eingebettete Systeme)
Prof. Dr. Dominik Henrich

EINVERSTÄNDNISERKLÄRUNG DES TEILNEHMERS

Dies ist von den Freiwilligen auszufüllen. Wir bitten Sie, die nachfolgenden Fragen sorgfältig zu lesen.

Haben Sie die verbale Einführung zu dieser Studie erhalten?	JA/NEIN
Haben Sie die Möglichkeit gehabt, Fragen zu stellen und über die Studie zu sprechen?	JA/NEIN
Haben Sie befriedigende Antworten auf alle Ihre Fragen erhalten?	JA/NEIN
Haben Sie genügend Informationen über diese Studie erhalten?	JA/NEIN
Sind Sie informiert, dass die Teilnahme an dieser Studie freiwillig ist und zu jedem Zeitpunkt und ohne Angabe von Gründen abgebrochen werden kann?	JA/NEIN
Sind Sie mit der Verarbeitung der in dieser Studie aufgenommenen personenbezogenen Daten zum Zweck der vorliegenden Studie einverstanden?	JA/NEIN
Sind Sie mit der Veröffentlichung folgender Daten in anonymisierter Form einverstanden: getrackte Positionen der Hand und Abarbeitungsreihenfolge samt Zeitinformationen (die Zustimmung ist optional und hilft dabei, das System in Simulation reproduzieren zu können)	JA/NEIN

Unterschrift……..……Datum……...

Name in Druckbuchstaben..……..……...

Die von uns gesammelten Informationen werden nie so abgespeichert, dass Personen identifiziert werden können. Die Informationen werden in zusammengefasster Form publiziert. Alle verbalen Äußerungen Ihrerseits werden in den Publikationen anonymisiert dargestellt. Sie können Ihr Einverständnis zur Verarbeitung der aufgenommenen personenbezogenen Daten jederzeit folgenlos und ohne Angabe von Gründen widerrufen. Sie haben jederzeit das Recht, uns aufzufordern, Ihre Daten aus unseren Datenbanken zu entfernen.

UNIVERSITÄT BAYREUTH

Lehrstuhl für Angewandte Informatik III
(Robotik und Eingebettete Systeme)
Prof. Dr. Dominik Henrich

PARTICIPANT INFORMED CONSENT

To be completed by volunteers. We would like you to read the following questions carefully.

Have you received the verbal introduction to this study?	YES/NO
Have you had an opportunity to ask questions and discuss this study?	YES/NO
Have you received satisfactory answers to all your questions?	YES/NO
Have you received enough information about this study?	YES/NO
Are you aware that participation in this study is voluntary and can be withdrawn at any time, without justification?	YES/NO
Do you consent with the use of the individual-related data recorded in this study for the purpose of the study at hand?	YES/NO
Do you consent with publishing following data in anonymized form: tracked hand positions and build order including time stamps (consent is optional but helps to reproduce the system in simulation)	YES/NO

Signed……..……**Date**……...

Name in block letters……..……...

Information that we collect will never be reported in a way that individuals can be identified. Information will be reported in aggregate, and any verbal comments that you make, if written about in subsequent papers, will be presented anonymously. You can revoke your consent to the use of your individual-related data at any time, without repercussions, without giving reasons. You have the right to request us to eliminate your data from our databases at any time.

Team Fluency

17. Human-Robot Fluency

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
The human-robot team worked fluently together.	○	○	○	○	○
The human-robot team's fluency improved over time.	○	○	○	○	○
The robot contributed to the fluency of the interaction.	○	○	○	○	○
I had to carry the weight to make the human-robot team better.	○	○	○	○	○
The robot contributed equally to the team performance.	○	○	○	○	○
I was the most important member on the team.	○	○	○	○	○
I trusted the robot to do the right thing at the right time.	○	○	○	○	○
The robot was trusthworthy.	○	○	○	○	○
The robot was intelligent.	○	○	○	○	○
The robot commited to the task.	○	○	○	○	○
The human-robot team improved over time.	○	○	○	○	○
The human-robot team's fluency improved over time.	○	○	○	○	○
The robot's performance improved over time.	○	○	○	○	○

18. Working Alliance

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I feel uncomfortable with the robot.	○	○	○	○	○
The robot and I understand each other.	○	○	○	○	○
I believe the robot likes me.	○	○	○	○	○
The robot and I respect each other.	○	○	○	○	○
I'm confident in the robot's ability to help me.	○	○	○	○	○
I feel that the robot appreciates me.	○	○	○	○	○
The robot and I trust each other.	○	○	○	○	○
The robot perceives accurately what my goals are.	○	○	○	○	○
The robot does not understand what I am trying to accomplish.	○	○	○	○	○
The robot and I are working towards mutually agreed upon goals.	○	○	○	○	○
I find what I am doing with the robot confusing.	○	○	○	○	○
The robot had an important contribution to the success of the team.	○	○	○	○	○
The robot was committed to the success of the team.	○	○	○	○	○
I was committed to the success of the team.	○	○	○	○	○
The robot was cooperative.	○	○	○	○	○

Zusammenspiel im Team

17. Mensch-Roboter Zusammenspiel

	Trifft nicht zu	Trifft eher nicht zu	teils-teils	Trifft eher zu	Trifft zu
Das Team aus Mensch und Roboter arbeitete flüssig zusammen.	◯	◯	◯	◯	◯
Das Zusammenspiel lief mit der Zeit flüssiger.	◯	◯	◯	◯	◯
Der Roboter trug zum flüssigen Zusammenspiel bei.	◯	◯	◯	◯	◯
Ich trug die Hauptlast, um das Team besser zu machen.	◯	◯	◯	◯	◯
Der Robter trug in gleichem Maße zur Teamleistung bei.	◯	◯	◯	◯	◯
Ich war das wichtigste Teammitglied.	◯	◯	◯	◯	◯
Ich vertraute dem Roboter das Richtige zur richten Zeit zu tun.	◯	◯	◯	◯	◯
Der Roboter war vertrauenswürdig.	◯	◯	◯	◯	◯
Der Roboter war intelligent.	◯	◯	◯	◯	◯
Der Roboter trug zur Aufgabe bei.	◯	◯	◯	◯	◯
Das Team hat sich mit die Zeit verbessert.	◯	◯	◯	◯	◯
Die Flüssigkeit der Zusammenarbeit hat sich mit der Zeit verbessert.	◯	◯	◯	◯	◯
Die Leistung des Roboters hat sich mit der Zeit gesteigert.	◯	◯	◯	◯	◯

18. Working Alliance

	Trifft nicht zu	Trifft eher nicht zu	teils-teils	Trifft eher zu	Trifft zu
Ich fühle mich mit dem Roboter unwohl.	◯	◯	◯	◯	◯
Der Roboter und ich verstehen sich.	◯	◯	◯	◯	◯
Ich glaube, der Roboter mag mich.	◯	◯	◯	◯	◯
Der Roboter und ich respektieren sich.	◯	◯	◯	◯	◯
Ich bin voll Zuversicht in die Fähigkeit des Roboters mir zu helfen.	◯	◯	◯	◯	◯
Ich glaube, der Roboter wertschätzt mich.	◯	◯	◯	◯	◯
Der Roboter und ich vertrauen uns.	◯	◯	◯	◯	◯
Der Roboter nimmt genau wahr, was meine Ziele sind.	◯	◯	◯	◯	◯
Der Roboter versteht nicht, was ich zu erreichen suche.	◯	◯	◯	◯	◯
Der Roboter und ich arbeiten auf Ziele zu, auf die wir uns geeinigt haben.	◯	◯	◯	◯	◯
Ich finde es verwirrend, was ich mit dem Roboter machen soll.	◯	◯	◯	◯	◯
Der Roboter lieferte einen wichtigen	◯	◯	◯	◯	◯

Dieser Inh … ıtümer

Final questionnaire

* Dieses Formular wird Ihren Namen aufzeichnen. Bitte tragen Sie Ihren Namen ein.

1. Participant ID

   Filled by supervisor

   Der Wert muss eine Zahl sein.

2. Did you notice differences in the robot's behaviour (comparing the three trials)?

   - ◯ Yes
   - ◯ No

Order the trials by each aspect. Put the trial first that matches the aspect best.

3. At the end of the trial the cooperation was fluent.

- Trial 1
- Trial 2
- Trial 3

4. I knew precisely how the robot would act.

- Trial 1
- Trial 2
- Trial 3

Further thoughts or comments?

5. (e.g. what the robot behaviours were, ...)

Demographic Information

6. Age

Der Wert muss eine Zahl sein.

7. Gender

- ◯ Woman
- ◯ Man
- ◯ Non-binary
- ◯ Prefer not to say

8. Handedness

- ◯ Left
- ◯ Right
- ◯ Both

9. Highest Academic Degree

- ◯ general qualification for university entrance
- ◯ Bachelor
- ◯ Master
- ◯ Other

10. Do you have sensorimotor impairements that had a negative impact on the study?

Sprache: Deutsch

Abschlussbefragung

* Dieses Formular wird Ihren Namen aufzeichnen. Bitte tragen Sie Ihren Namen ein.

1. Participant ID

   Filled by supervisor

   Der Wert muss eine Zahl sein.

2. Haben Sie einen Unterschied im Roboterverhalten bemerkt (wenn sie die drei Durchläufe vergleichen)?

   - ◯ Ja
   - ◯ Nein

Sortieren Sie die Durchläufe anhand des genannten Aspektes. Ziehen Sie denjenigen nach oben, auf den es am besten zutrifft.

3. Am Ende des Durchgangs verlief die Zusammenarbeit flüssig.

- Durchlauf 1
- Durchlauf 2
- Durchlauf 3

4. Ich wusste genau, wie sich der Roboter verhält.

- Durchlauf 1
- Durchlauf 2
- Durchlauf 3

Weitere Gedanken oder Anmerkungen?

5. (z.B. was die Roboterverhalten waren, ...)

Demographische Informationen

6. Alter

Der Wert muss eine Zahl sein.

7. Geschlecht

- ◯ Weiblich
- ◯ Männlich
- ◯ Divers
- ◯ Keine Angabe

8. Händigkeit

- ◯ Linkshänder
- ◯ Rechtshänder
- ◯ Beides

9. Höchster Bildungsabschluss

- ◯ Hochschulzulassung
- ◯ Bachelor
- ◯ Master
- ◯ Sonstiges

10. Haben Sie sensomotorische Einschränkungen, die einen negativen Effekt auf die Studie hatten?

B.3. Evaluation

B.3.1. Statistical Tests

Scale	Comparison	One-sided Wilcoxon signed-rank test	Hodges-Lehmann-Sen shift estimator
Fluency (short)	FIXED vs. ADAP.	0.7741	-0.125 within [-0.5, 0.25]
Fluency (short)	FIXED vs. ADV.	2.408e-05	-0.625 within [-1.25, -0.125]
Fluency (short)	ADAP. vs. ADV.	0.002925	-0.5 within [-1.125, 0]
Fluency (subscale)	FIXED vs. ADAP.	0.4279	0 within [-0.3333, 0.6667]
Fluency (subscale)	FIXED vs. ADV.	0.0004353	-0.6667 within [-1.333, 0]
Fluency (subscale)	ADAP. vs. ADV.	0.001007	-1 within [-1.333, -0.3333]
Contribution	FIXED vs. ADAP.	0.6913	0 within [-0.5, 0.5]
Contribution	FIXED vs. ADV.	0.0299	-0.25 within [-0.75, 0.25]
Contribution	ADAP. vs. ADV.	0.03936	-0.25 within [-0.75, 0.25]
Trust	FIXED vs. ADAP.	0.808	0 within [-0.6667, 0.3333]
Trust	FIXED vs. ADV.	0.001623	-0.6667 within [-1, 0]
Trust	ADAP. vs. ADV.	0.01438	-0.3333 within [-1, 0.3333]
Teammate Traits	FIXED vs. ADAP.	0.8596	-0.25 within [-0.75, 0.25]
Teammate Traits	FIXED vs. ADV.	2.444e-05	-0.75 within [-1.25, -0.25]
Teammate Traits	ADAP. vs. ADV.	0.001866	-0.5 within [-1, 0]
Improvement	FIXED vs. ADAP.	0.687	0 within [-0.6667, 0.6667]
Improvement	FIXED vs. ADV.	0.0008507	-0.6667 within [-1.333, 0]
Improvement	ADAP. vs. ADV.	0.01243	-0.6667 within [-1.333, 0]
Working Alliance	FIXED vs. ADAP.	0.8619	0 within [-0.7143, 0.4286]
Working Alliance	FIXED vs. ADV.	0.002163	-0.2857 within [-0.8571, 0.1429]
Working Alliance	ADAP. vs. ADV.	0.01509	-0.2857 within [-0.9524, 0.2857]
Goal perception	FIXED vs. ADAP.	0.2743	0 within [-0.6667, 0.6667]
Goal perception	FIXED vs. ADV.	0.1175	0 within [-1, 0.6667]
Goal perception	ADAP. vs. ADV.	0.04197	-0.3333 within [-1, 0.3333]
Individual Measures	FIXED vs. ADAP.	0.7185	0 within [-0.5, 0.25]
Individual Measures	FIXED vs. ADV.	0.0008996	-0.25 within [-0.75, 0.25]
Individual Measures	ADAP. vs. ADV.	0.01143	-0.25 within [-0.75, 0.25]

B.3.2. Human Activity Calculation

Position and hand movement are used as proxies to estimate human activity. Hand motion is recorded by the HoloLens 2 and sent to the robot. If the HoloLens 2 is not able to detect one of the hands, the algorithm described in Chapter 4 continues the tracking. The data is then saved into a log file using the timestamps from the robot system. Invalid lines, that originated from race conditions when writing the log file, are dropped. From the hand pose, only the position of the wrist is used for the following analysis. The workspace (refer to Figure 7.1 and Figure B.2) is divided into three areas based on the robot coordinate system. All areas are unlimited in their y and z dimensions. In the area $10\,\mathrm{cm} \leq x \leq 70\,\mathrm{cm}$, the human is assumed to be always active. In the area $70\,\mathrm{cm} \leq x \leq 90\,\mathrm{cm}$, the participant can be active or idle. The distinction is based on the velocity of the hand. The remaining area is the idling area. To estimate the movement speed of the wrist, the high-frequency noise of the position data needs to be removed. To achieve that, the unevenly spaced time series is resampled at a rate of $10\,\mathrm{Hz}$ using a sliding averaging window of size $400\,\mathrm{ms}$. The resampling is done on each dimension independently. Speed is calculated as the difference between two consecutive positions. If the speed exceeds $0.05\,\mathrm{m\,s^{-1}}$ and the hand is within the middle area, the user is assumed to be active. The threshold was manually tuned by picking active and idle intervals[7] from the logging data and plotting the wrist speed. Afterwards, a plausibility check was run by visually inspecting the Gantt charts of human activity, idling, and action completion for all trials. Intervals for activity and idling are formed by dividing the timespan into intervals of one second. If the human is active at one data point in the one-second interval, the interval is added to the active time; otherwise, to the idle time. Afterwards, a plausibility check was run by visually inspecting the Gantt charts of human activity, idling, and action completion for all trials. Despite the effort, some rare situations are not be identified correctly, e.g. if the participant is moving the hand within the middle area without being active.

[7] Those intervals are found by turning the keypoints into an animation sequence in blender

List of Tables

List of Figures

Bibliography

[1] Heiner Lasi et al. 'Industry 4.0'. In: *Business & Information Systems Engineering* 6.4 (2014), pp. 239–242. DOI: 10.1007/s12599-014-0334-4.

[2] Manuel Sanchez, Ernesto Exposito and Jose Aguilar. 'Industry 4.0: survey from a system integration perspective'. In: *International Journal of Computer Integrated Manufacturing* 33.10–11 (2020), pp. 1017–1041. DOI: 10.1080/0951192x.2020.1775295.

[3] Mohd Aiman Kamarul Bahrin et al. 'Industry 4.0: A review on industrial automation and robotic'. In: *Jurnal Teknologi* 78.6–13 (2016). DOI: https://doi.org/10.11113/jt.v78.9285.

[4] Ruchi Goel and Pooja Gupta. 'Robotics and Industry 4.0'. In: *A Roadmap to Industry 4.0: Smart Production, Sharp Business and Sustainable Development*. Springer International Publishing, 2019, pp. 157–169. DOI: 10.1007/978-3-030-14544-6_9.

[5] International Federation of Robotics via AI Index – with minor processing by Our World in Data. *Annual industrial robots installed*. https://ourworldindata.org/grapher/industrial-robots-annual-installations-total-operational. [Online; accessed 2025-01-16]. 2024.

[6] Kadir Alpaslan Demir, Gözde Döven and Bülent Sezen. 'Industry 5.0 and Human-Robot Co-working'. In: *Procedia Computer Science* 158 (2019), pp. 688–695. DOI: 10.1016/j.procs.2019.09.104.

[7] Wim Lambrechts et al. 'Human Factors Influencing the Implementation of Cobots in High Volume Distribution Centres'. In: *Logistics* 5.2 (2021), p. 32. DOI: 10.3390/logistics5020032.

[8] Ikrom Kambarov et al. 'From Human to Robot Interaction towards Human to Robot Communication in Assembly Systems'. In: *The Eurasia Proceedings of Science Technology Engineering and Mathematics* 23 (2023), pp. 241–252. DOI: 10.55549/epstem.1365802.

[9] Afonso Amaral and Paulo Peças. 'SMEs and Industry 4.0: Two case studies of digitalization for a smoother integration'. In: *Computers in Industry* 125 (2021), p. 103333. DOI: 10.1016/j.compind.2020.103333.

[10] Omolayo M. Ikumapayi et al. 'Human-Robot Co-working Improvement via Revolutionary Automation and Robotic Technologies – An overview'. In: *Procedia Computer Science* 217 (2023), pp. 1345–1353. DOI: 10.1016/j.procs.2022.12.332.

[11] Figure. *Figure Status Update - Real World Task*. https://www.youtube.com/watch?v=gEjXcEU3Bbw. [Online; accessed 2024-06-14]. 2024.

[12] Wandelbots. *Sanding*. https://www.wandelbots.com/sanding. [Online; accessed 2024-06-14]. 2024.

[13] Kevin Black et al. *π_0: A Vision-Language-Action Flow Model for General Robot Control*. 2024. DOI: 10.48550/ARXIV.2410.24164.

[14] Lihui Wang et al. 'Symbiotic human-robot collaborative assembly'. In: *CIRP annals* 68.2 (2019), pp. 701–726. DOI: 10.1016/j.cirp.2019.05.002.

[15] Pietro Zanuttigh et al. *Time-of-Flight and Structured Light Depth Cameras*. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-30973-6.

[16] Anthony Favier, Shashank Shekhar and Rachid Alami. 'Models and Algorithms for Human-Aware Task Planning with Integrated Theory of Mind'. In: *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2023. DOI: 10.1109/ro-man57019.2023.10309437.

[17] Dominik Riedelbauch. 'Dynamic Task Sharing for Flexible Human-Robot Teaming under Partial Workspace Observability'. en. Dissertation. Bayreuth: Universität Bayreuth, 2020. DOI: 10.15495/EPUB_UBT_00005156.

[18] D. Olsen and M. Goodrich. 'Metrics for evaluating human-robot interactions'. In: *4th International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*. Maryland, 2003.

[20] Daniel M. Oppenheimer. 'The secret life of fluency'. In: *Trends in Cognitive Sciences* 12.6 (2008), pp. 237–241. DOI: 10.1016/j.tics.2008.02.014.

[21] Guy Hoffman. 'Evaluating Fluency in Human–Robot Collaboration'. In: *IEEE Transactions on Human-Machine Systems* 49.3 (2019), pp. 209–218. DOI: 10.1109/thms.2019.2904558.

[22] Elliot E. Entin and Daniel Serfaty. 'Adaptive Team Coordination'. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 41.2 (1999), pp. 312–325. DOI: 10.1518/001872099779591196.

[23] Guy Hoffman and Cynthia Lynn Breazeal. 'Anticipatory Perceptual Simulation for Human-Robot Joint Practice: Theory and Application Study'. In: *AAAI Conference on Artificial Intelligence*. 2008.

[24] Matthew Gombolay et al. 'Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences'. In: *The International Journal of Robotics Research* 36.5-7 (2017), pp. 597–617. DOI: 10.1177/0278364916688255.

[25] Friedhelm Nachreiner. 'Standards for ergonomics principles relating to the design of work systems and to mental workload'. In: *Applied Ergonomics* 26.4 (1995), pp. 259–263. DOI: 10.1016/0003-6870(95)00029-C.

[26] Christina Schmidbauer et al. 'An Empirical Study on Workers' Preferences in Human–Robot Task Assignment in Industrial Assembly Systems'. In: *IEEE Transactions on Human-Machine Systems* 53.2 (2023), pp. 293–302. DOI: 10.1109/thms.2022.3230667.

[27] Valeria Villani et al. 'Survey on Human-Robot Interaction for Robot Programming in Industrial Applications'. In: *IFAC-PapersOnLine* 51.11 (2018), pp. 66–71. DOI: 10.1016/j.ifacol.2018.08.236.

[28] Natalie Sebanz, Harold Bekkering and Günther Knoblich. 'Joint action: bodies and minds moving together'. In: *Trends in Cognitive Sciences* 10.2 (2006), pp. 70–76. DOI: 10.1016/j.tics.2005.12.009.

[29] Bilge Mutlu, Allison Terrell and Chien-Ming Huang. 'Coordination mechanisms in human-robot collaboration'. In: *ACM/IEEE Intl. Conf. on Human-Robot Interaction (HRI)-Workshop on Collaborative Manipulation*. 2013, pp. 1–6.

[30] Andrea Bauer, Dirk Wollherr and Martin Buss. 'Human–Robot Collaboration: a Survey'. In: *International Journal of Humanoid Robotics* 05.01 (2008), pp. 47–66. DOI: 10.1142/s0219843608001303.

[31] Brian Gleeson et al. 'Gestures for industry: intuitive human-robot communication from human observation'. In: *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*. IEEE Press. 2013, pp. 349–356. DOI: 10.1109/hri.2013.6483609.

[32] Kenneth James Williams Craik. *The nature of explanation*. Vol. 445. CUP Archive, 1967.

[33] Renée J. Stout et al. 'Planning, Shared Mental Models, and Coordinated Performance: An Empirical Link Is Established'. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 41.1 (1999), pp. 61–71. DOI: 10.1518/001872099779577273.

[34] Aaquib Tabrez, Matthew B. Luebbers and Bradley Hayes. 'A Survey of Mental Modeling Techniques in Human–Robot Teaming'. In: *Current Robotics Reports* 1.4 (2020), pp. 259–267. DOI: 10.1007/s43154-020-00019-0.

[35] Fangyun Zhao, Curt Henrichs and Bilge Mutlu. 'Task Interdependence in Human-Robot Teaming'. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020. DOI: 10.1109/ro-man47096.2020.9223555.

[36] Diana I Tamir, Mark A Thornton and Diana I Tamir. 'Predicting other people shapes the social mind'. In: *Advances in Experimental Social Psychology*. Academic Press Inc., 2024, pp. 263–315. DOI: 10.1016/bs.aesp.2023.11.003.

[37] Sachiko Matsumoto and Laurel D Riek. 'Fluent coordination in proximate human robot teaming'. In: *In Proceedings of the Robotics, Science, and Systems (RSS) Workshop on AI and Its Alternatives for Shared Autonomy in Assistive and Collaborative Robotics*. 2019.

[38] Alessandro Roncone, Olivier Mangin and Brian Scassellati. 'Transparent role assignment and task allocation in human robot collaboration'. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1014–1021. DOI: 10.1109/icra.2017.7989122.

[39] Heramb Nemlekar et al. 'Two-Stage Clustering of Human Preferences for Action Prediction in Assembly Tasks'. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021. DOI: 10.1109/icra48506.2021.9561649.

[40] Claus Lenz. 'Context-aware human-robot collaboration as a basis for future cognitive factories'. Dissertation. München: Technische Universität München, 2011.

[41] Kelsey P. Hawkins et al. 'Anticipating human actions for collaboration in the presence of task and sensor uncertainty'. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014. DOI: 10.1109/icra.2014.6907165.

[42] Begerowski S.R. et al. 'Examining the Effects of Cognitive Assistive Agents on Team Coordination in Manufacturing Teams'. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 66.1 (2022), pp. 1184–1188. DOI: 10.1177/1071181322661401.

[43] Matthew C. Gombolay et al. 'Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams'. In: *Autonomous Robots* 39.3 (2015), pp. 293–312. DOI: 10.1007/s10514-015-9457-9.

[44] Audun Rønning Sanderud. 'Responsible robots: a novel approach to safe and productive human-robot collaboration'. PhD thesis. Chuo University, 2016.

[45] Brenda Castro et al. 'Who takes the lead? Automated scheduling for human-robot teams'. In: *2017 AAAI Fall Symposium Series*. 2017.

[46] Hema S. Koppula, Ashesh Jain and Ashutosh Saxena. 'Anticipatory Planning for Human-Robot Teams'. In: *Springer Tracts in Advanced Robotics*. Springer International Publishing, 2015, pp. 453–470. DOI: 10.1007/978-3-319-23778-7_30.

[47] Huaijiang Zhu, Volker Gabler and Dirk Wollherr. 'Legible action selection in human-robot collaboration'. In: *26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE. 2017, pp. 354–359. DOI: 10.1109/roman.2017.8172326.

[48] Moritz C Buehler and Thomas H Weisswange. 'Online inference of human belief for cooperative robots'. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 409–415. DOI: 10.1109/iros.2018.8594076.

[49] Steven James Levine and Brian Charles Williams. 'Concurrent Plan Recognition and Execution for Human-Robot Teams.' In: *Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS)*. 2014, pp. 490–498. DOI: 10.1609/icaps.v24i1.13672.

[50] Alberto Gottardi et al. 'Dynamic Human-Aware Task Planner for Human-Robot Collaboration in Industrial Scenario'. In: *2023 European Conference on Mobile Robots (ECMR)*. IEEE, 2023. DOI: 10.1109/ecmr59166.2023.10256268.

[51] Christoph Petzoldt et al. 'Implementation and Evaluation of Dynamic Task Allocation for Human–Robot Collaboration in Assembly'. In: *Applied Sciences* 12.24 (2022), p. 12645. DOI: 10.3390/app122412645.

[52] Costanza Messeri et al. 'A Dynamic Task Allocation Strategy to Mitigate the Human Physical Fatigue in Collaborative Robotics'. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2178–2185. DOI: 10.1109/lra.2022.3143520.

[53] Jimmy Baraglia et al. 'Efficient human-robot collaboration: When should a robot take initiative?' In: *The International Journal of Robotics Research* 36.5-7 (2017), pp. 563–579. DOI: 10.1177/0278364916688253.

[54] Edoardo Lamon et al. *A Unified Architecture for Dynamic Role Allocation and Collaborative Task Planning in Mixed Human-Robot Teams*. Tech. rep. 2023. DOI: 10.48550/ARXIV.2301.08038.

[55] Stefanos Nikolaidis et al. 'Improved human–robot team performance through cross-training, an approach inspired by human team training practices'. In: *The International Journal of Robotics Research* 34.14 (2015), pp. 1711–1730. DOI: 10.1177/0278364915609673.

[56] Heramb Nemlekar et al. 'Transfer Learning of Human Preferences for Proactive Robot Assistance in Assembly Tasks'. In: *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. HRI '23. ACM, 2023. DOI: 10.1145/3568162.3576965.

[57] Min Chen et al. 'Planning with trust for human-robot collaboration'. In: *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. ACM. 2018, pp. 307–315. DOI: 10.1145/3171221.3171264.

[58] Crystal Chao and Andrea Thomaz. 'Timing in Multimodal Turn-Taking Interactions: Control and Analysis Using Timed Petri Nets'. In: *Journal of Human-Robot Interaction* 1.1 (2012), pp. 4–25. DOI: 10.5898/jhri.1.1.chao.

[59] Crystal Chao and Andrea Thomaz. 'Timed Petri nets for fluent turn-taking over multimodal interaction resources in human-robot collaboration'. In: *The International Journal of Robotics Research* 35.11 (2016), pp. 1330–1353. DOI: 10.1177/0278364915627291.

[60] Anthony Favier and Rachid Alami. 'Planning Concurrent Actions and Decisions in Human-Robot Joint Action Context'. In: *HRI'24 Workshop Symbiotic Society with Avatars (SSA)*. 2024.

[61] Riccardo Gervasi et al. 'User Experience and Physiological Response in Human-Robot Collaboration: A Preliminary Investigation'. In: *Journal of Intelligent and Robotic Systems* 106.2 (2022). DOI: 10.1007/s10846-022-01744-8.

[62] Przemyslaw A. Lasota and Julie A. Shah. 'Analyzing the Effects of Human-Aware Motion Planning on Close-Proximity Human-Robot Collaboration'. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 57.1 (2015), pp. 21–33. DOI: 10.1177/0018720814565188.

[63] Julie Shah et al. 'Improved human-robot team performance using chaski, a human-inspired plan execution system'. In: *Proceedings of the 6th international conference on Human-robot interaction*. ACM. 2011, pp. 29–36. DOI: 10.1145/1957656.1957668.

[64] Athanasios C. Tsitos and Maria Dagioglou. *Enhancing team performance with transfer-learning during real-world human-robot collaboration*. 2022. DOI: 10.48550/ARXIV.2211.13070.

[65] John Brooke. 'SUS: a retrospective'. In: *Journal of usability studies* 8.2 (2013), pp. 29–40.

[66] Kristin E. Schaefer. 'Measuring Trust in Human Robot Interactions: Development of the "Trust Perception Scale-HRI"'. In: *Robust Intelligence and Trust in Autonomous Systems*. Springer US, 2016, pp. 191–218. DOI: 10.1007/978-1-4899-7668-0_10.

[67] Sandra Devin et al. 'Evaluating the pertinence of robot decisions in a human-robot joint action context: The PeRDITA questionnaire'. In: *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE. 2018, pp. 144–151. DOI: 10.1109/roman.2018.8525785.

[68] Sandra G. Hart and Lowell E. Staveland. 'Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research'. In: *Advances in Psychology* 52 (1988), pp. 139–183. DOI: 10.1016/S0166-4115(08)62386-9.

[70] Julie Shah and Cynthia Breazeal. 'An Empirical Analysis of Team Coordination Behaviors and Action Planning With Application to Human-Robot Teaming'. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 52.2 (2010). Ed. by Jamie C. Gorman, Nancy J. Cooke and Eduardo Salas, pp. 234–245. DOI: 10.1177/0018720809350882.

[71] Brian Gleeson et al. 'Human-robot communication for collaborative assembly'. In: *GRAND Symposium*. 2013.

[72] Junming Fan, Pai Zheng and Shufei Li. 'Vision-based holistic scene understanding towards proactive human–robot collaboration'. In: *Robotics and Computer-Integrated Manufacturing* 75 (2022), p. 102304. DOI: 10.1016/j.rcim.2021.102304.

[73] Mark A. Thornton and Diana I. Tamir. 'People accurately predict the transition probabilities between actions'. In: *Science Advances* 7.9 (2021). DOI: 10.1126/sciadv.abd4995.

[74] Jinbao Wang et al. 'Deep 3D human pose estimation: A review'. In: *Computer Vision and Image Understanding* 210 (2021), p. 103225. DOI: 10.1016/j.cviu.2021.103225.

[75] Zhe Cao et al. 'OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (2021), pp. 172–186. DOI: 10.1109/tpami.2019.2929257.

[76] Camillo Lugaresi et al. *MediaPipe: A Framework for Building Perception Pipelines*. 2019. DOI: 10.48550/ARXIV.1906.08172.

[77] Michal Tölgyessy, Martin Dekan and Ľuboš Chovanec. 'Skeleton Tracking Accuracy and Precision Evaluation of Kinect V1, Kinect V2, and the Azure Kinect'. In: *Applied Sciences* 11.12 (2021), p. 5756. DOI: 10.3390/app11125756.

[78] Ali Erol et al. 'Vision-based hand pose estimation: A review'. In: *Computer Vision and Image Understanding* 108.1-2 (2007), pp. 52–73. DOI: 10.1016/j.cviu.2006.10.012.

[79] Wikipedia contributors. *Leap Motion — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Leap_Motion&oldid=1238141749. [Online; accessed 2024-08-28]. 2024.

[80] Microsoft. *Microsoft HoloLens | Mixed Reality Technology for Business*. https://web.archive.org/web/20230127133550/https://www.microsoft.com/en-us/HoloLens. [Online; accessed 2023-07-27]. 2023.

[81] Meta. *Meta Quest 3: New mixed reality VR headset – Shop now | Meta Store*. https://www.meta.com/en/quest/quest-3/. [Online; accessed 2024-08-28]. 2024.

[82] Jože Guna et al. 'An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking'. In: *Sensors* 14.2 (2014), pp. 3702–3720. DOI: 10.3390/s140203702.

[83] Inês Soares et al. 'Accuracy and Repeatability Tests on HoloLens 2 and HTC Vive'. In: *Multimodal Technologies and Interaction* 5.8 (2021), p. 47. DOI: 10.3390/mti5080047.

[84] Rui Li, Zhenyu Liu and Jianrong Tan. 'A survey on 3D hand pose estimation: Cameras, methods, and datasets'. In: *Pattern Recognition* 93 (2019), pp. 251–272. DOI: 10.1016/j.patcog.2019.04.026.

[85] Ayan Sinha, Chiho Choi and Karthik Ramani. 'DeepHand: Robust Hand Pose Estimation by Completing a Matrix Imputed with Deep Features'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. DOI: 10.1109/cvpr.2016.450.

[86] Chi Xu et al. 'Lie-X: Depth Image Based Articulated Object Pose Estimation, Tracking, and Action Recognition on Lie Groups'. In: *International Journal of Computer Vision* 123.3 (2017), pp. 454–478. DOI: 10.1007/s11263-017-0998-6.

[87] Kha Gia Quach et al. 'Depth-based 3D hand pose tracking'. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016. DOI: 10.1109/icpr.2016.7900051.

[88] Anastasia Tkach, Mark Pauly and Andrea Tagliasacchi. 'Sphere-meshes for real-time hand modeling and tracking'. In: *ACM Transactions on Graphics* 35.6 (2016), pp. 1–11. DOI: 10.1145/2980179.2980226.

[89] Iason Oikonomidis, Nikolaos Kyriazis and Antonis A Argyros. 'Efficient model-based 3D tracking of hand articulations using Kinect.' In: *BmVC*. Vol. 1. 2. 2011, p. 3. DOI: 10.5244/c.25.101.

[90] Tzu-Yang Chen et al. 'Learning a deep network with spherical part model for 3D hand pose estimation'. In: *Pattern Recognition* 80 (2018), pp. 1–20. DOI: 10.1016/j.patcog.2018.02.029.

[91] Zhengyou Zhang. 'Camera Parameters (Intrinsic, Extrinsic)'. In: *Computer Vision*. Springer International Publishing, 2021, pp. 135–140. DOI: 10.1007/978-3-030-63416-2_152.

[92] Radu Bogdan Rusu and Steve Cousins. '3d is here: Point cloud library (pcl)'. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 1–4. DOI: 10.1109/icra.2011.5980567.

[93] Antonis A. Argyros and Manolis I. A. Lourakis. 'Real-Time Tracking of Multiple Skin-Colored Objects with a Possibly Moving Camera'. In: *Computer Vision - ECCV 2004*. Springer Berlin Heidelberg, 2004, pp. 368–379. DOI: 10.1007/978-3-540-24672-5_29.

[94] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[95] Konrad P Cop et al. 'New Metrics for Industrial Depth Sensors Evaluation for Precise Robotic Applications'. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021. DOI: 10.1109/iros51168.2021.9636322.

[96] Hamid Rezatofighi et al. 'Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. DOI: 10.1109/cvpr.2019.00075.

[97] Fai Chen Chen et al. 'Constraint Study for a Hand Exoskeleton: Human Hand Kinematics and Dynamics'. In: *Journal of Robotics* 2013 (2013), pp. 1–17. DOI: 10.1155/2013/910961.

[98] Irene Albrecht, Jörg Haber and Hans-Peter Seidel. 'Construction and animation of anatomically based human hand models'. In: *Symposium on Computer Animation* (2003), pp. 98–109. DOI: 10.2312/SCA03/098-109.

[99] Franziska Mueller et al. 'Ganerated hands for real-time 3d hand tracking from monocular rgb'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 49–59. DOI: 10.1109/cvpr.2018.00013.

[100] J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005.

[101] Tomas Simon et al. 'Hand keypoint detection in single images using multiview bootstrapping'. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 1145–1153. DOI: 10.1109/cvpr.2017.494.

[102] W. Kabsch. 'A solution for the best rotation to relate two sets of vectors'. In: *Acta Crystallographica Section A* 32.5 (1976), pp. 922–923. DOI: 10.1107/s0567739476001873.

[103] Julia. *Average Finger Size: Find Out Your Finger Size*. https://size-charts.com/topics/body-size-chart/average-finger-size-find-out-your-finger-size/. [Online, accessed 2025-02-24]. 2022.

[105] Allah Bux, Plamen Angelov and Zulfiqar Habib. 'Vision Based Human Activity Recognition: A Review'. In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2016, pp. 341–371. DOI: 10.1007/978-3-319-46562-3_23.

[106] Selim Erol and Philipp Hold. 'Keeping Track of the Physical in Assembly Processes'. In: *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE, 2016. DOI: 10.1109/edocw.2016.7584365.

[107] Andrea Zanchettin et al. 'Prediction of human activity patterns for human-robot collaborative assembly tasks'. In: *IEEE Transactions on Industrial Informatics* (2018). DOI: 10.1109/tii.2018.2882741.

[108] Kelsey P Hawkins et al. 'Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration'. In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. 2013, pp. 499–506. DOI: 10.1109/humanoids.2013.7030020.

[109] Julia Berg et al. 'Action Recognition in Assembly for Human-Robot-Cooperation using Hidden Markov Models'. In: *Procedia CIRP* 76 (2018), pp. 205–210. DOI: 10.1016/j.procir.2018.02.029.

[110] A. Sharma, P. K. Singh and P. Khurana. 'Analytical review on object segmentation and recognition'. In: *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*. 2016, pp. 524–530. DOI: 10.1109/CONFLUENCE.2016.7508176.

[111] Zhengxia Zou et al. 'Object Detection in 20 Years: A Survey'. In: *Proceedings of the IEEE* 111.3 (2019), pp. 257–276. DOI: 10.1109/jproc.2023.3238524.

[112] Adrien Kaiser, Jose Alonso Ybanez Zepeda and Tamy Boubekeur. 'A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data'. In: *Computer Graphics Forum* 38.1 (2018), pp. 167–196. DOI: 10.1111/cgf.13451.

[113] Sebastian Pimminger and Werner Kurschl. 'Prototyping Assistive Systems for Manual Assembly in Real Production Environments: Challenges and Lessons Learned'. In: *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*. IEEE, 2021. DOI: 10.1109/ichms53169.2021.9582461.

[114] Martina Pelosi et al. 'Siamese Network for assembly step recognition and quality assessment for Human-Robot Collaboration'. In: *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2024, pp. 3811–3817. DOI: 10.1109/case59546.2024.10711711.

[115] Jiazhen Pang, S.K. Ong and A.Y.C. Nee. 'Image and model sequences matching for on-site assembly stage identification'. In: *Robotics and Computer-Integrated Manufacturing* 72 (2021), p. 102185. DOI: 10.1016/j.rcim.2021.102185.

[116] Chengjun Chen et al. 'Monitoring of Assembly Process Using Deep Learning Technology'. In: *Sensors* 20.15 (2020), p. 4208. DOI: 10.3390/s20154208.

[117] Sebastian Pimminger et al. 'Low-cost tracking of assembly tasks in industrial environments'. In: *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2019. DOI: 10.1145/3316782.3321526.

[118] Kaziwa Saleh, Sandor Szenasi and Zoltan Vamossy. 'Occlusion Handling in Generic Object Detection: A Review'. In: *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2021. DOI: 10.1109/sami50585.2021.9378657.

[119] Nico Blodow et al. 'Perception and probabilistic anchoring for dynamic world state logging'. In: *2010 10th IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2010, pp. 160–166. DOI: 10.1109/ichr.2010.5686341.

[120] Pedro Zuidberg Dos Martires et al. 'Symbolic Learning and Reasoning With Noisy Data for Probabilistic Anchoring'. In: *Frontiers in Robotics and AI* 7 (2020). DOI: 10.3389/frobt.2020.00100.

[121] Jingguo Xue, Xueliang Hou and Ying Zeng. 'Review of Image-Based 3D Reconstruction of Building for Automated Construction Progress Monitoring'. In: *Applied Sciences* 11.17 (2021), p. 7840. DOI: 10.3390/app11177840.

[122] John Oyekan et al. 'Utilising low cost RGB-D cameras to track the real time progress of a manual assembly sequence'. In: *Assembly Automation* 40.6 (2019), pp. 925–939. DOI: 10.1108/aa-06-2018-078.

[123] Shane Gilroy, Edward Jones and Martin Glavin. 'Overcoming Occlusion in the Automotive Environment—A Review'. In: *IEEE Transactions on Intelligent Transportation Systems* 22.1 (2021), pp. 23–35. DOI: 10.1109/tits.2019.2956813.

[124] Dominik Riedelbauch, Tobias Werner and Dominik Henrich. 'Supporting a Human-Aware World Model Through Sensor Fusion'. In: *Advances in Service and Industrial Robotics*. Springer International Publishing, 2017, pp. 665–672. DOI: 10.1007/978-3-319-61276-8_70.

[125] Leslie Pack Kaelbling, Michael L Littman and Anthony R Cassandra. 'Planning and acting in partially observable stochastic domains'. In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134. DOI: 10.1016/s0004-3702(98)00023-x.

[126] Tobias Werner et al. 'ENACT: An Efficient and Extensible Entity-Actor Framework for Modular Robotics Software Components'. In: *47th International Symposium on Robotics (ISR)* (2016).

[127] Martijn Cramer, Karel Kellens and Eric Demeester. *APLAN: open assembly planning framework in FreeCAD*. 2024. DOI: https://doi.org/10.1016/j.procir.2024.07.011.

[128] C. Schuldt, I. Laptev and B. Caputo. 'Recognizing human actions: a local SVM approach'. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* IEEE, 2004. DOI: 10.1109/icpr.2004.1334462.

[129] M S Ryoo and J K Aggarwal. 'Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities'. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009. DOI: 10.1109/iccv.2009.5459361.

[130] Hilde Kuehne, Ali Arslan and Thomas Serre. 'The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014. DOI: 10.1109/cvpr.2014.105.

[131] Sebastian Stein and Stephen J. McKenna. 'Combining embedded accelerometers with computer vision for recognizing food preparation activities'. In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. UbiComp '13. ACM, 2013. DOI: 10.1145/2493432.2493482.

[132] Philipp Kratzer et al. 'MoGaze: A Dataset of Full-Body Motions that Includes Workspace Geometry and Eye-Gaze'. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 367–373. DOI: 10.1109/lra.2020.3043167.

[133] Yizhak Ben-Shabat et al. 'The IKEA ASM Dataset: Understanding People Assembling Furniture through Actions, Objects and Pose'. In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2021, pp. 846–858. DOI: 10.1109/wacv48630.2021.00089.

[134] Hema Swetha Koppula, Rudhir Gupta and Ashutosh Saxena. 'Learning human activities and object affordances from RGB-D videos'. In: *The International Journal of Robotics Research* 32.8 (2013). DOI: 10.1177/0278364913478446.

[135] Yansong Tang et al. 'COIN: A Large-Scale Dataset for Comprehensive Instructional Video Analysis'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. DOI: 10.1109/cvpr.2019.00130.

[136] Jean-Baptiste Alayrac et al. 'Unsupervised Learning from Narrated Instruction Videos'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. DOI: 10.1109/cvpr.2016.495.

[137] Boudewijn van Dongen. *BPI Challenge 2012*. nl. https://data.4tu.nl/articles/_/12689204/1. 2012. DOI: 10.4121/UUID:3926DB30-F712-4394-AEBC-75976070E91F.

[138] Mirko Polato. *Dataset belonging to the help desk log of an Italian Company*. en. https://data.4tu.nl/articles/_/12675977/1. 2017. DOI: 10.4121/UUID:0C60EDF1-6F83-4E75-9367-4C63B3E9D5BB.

[139] Takehiko Ohkawa et al. 'AssemblyHands: Towards egocentric activity understanding via 3d hand pose estimation'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12999–13008. DOI: 10.1109/cvpr52729.2023.01249.

[140] Naoya Yoshimura et al. 'OpenPack: A Large-Scale Dataset for Recognizing Packaging Works in IoT-Enabled Logistic Environments'. In: *2024 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2024. DOI: 10.1109/percom59722.2024.10494448.

[141] Yu Kong and Yun Fu. 'Human Action Recognition and Prediction: A Survey'. In: *International Journal of Computer Vision* 130.5 (2018), pp. 1366–1401. DOI: 10.1007/s11263-022-01594-9.

[142] Murchana Baruah, Bonny Banerjee and Atulya K. Nagar. 'Intent Prediction in Human–Human Interactions'. In: *IEEE Transactions on Human-Machine Systems* 53.2 (2023), pp. 458–463. DOI: 10.1109/thms.2023.3239648.

[143] Elisheva Bonchek-Dokow and Gal A. Kaminka. 'Towards computational models of intention detection and intention prediction'. In: *Cognitive Systems Research* 28 (2014), pp. 44–79. DOI: 10.1016/j.cogsys.2013.07.004.

[144] Sebastian Krusche et al. 'A new AI-based approach for contextualization and prediction of human activities in industrial robot applications.' In: *Tagungsband des 7. Kongresses Montage Handhabung Industrieroboter*. 2023. DOI: 10.1007/978-3-031-74010-7_19.

[145] Stefanos Nikolaidis et al. 'Efficient Model Learning from Joint-Action Demonstrations for Human-Robot Collaborative Tasks'. In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. HRI '15. ACM, 2015. DOI: 10.1145/2696454.2696455.

[146] Federico Formica et al. 'Neural Networks based Human Intent Prediction for Collaborative Robotics Applications'. In: *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021. DOI: 10.1109/icar53236.2021.9659328.

[147] Riccardo Bovo et al. 'Detecting errors in pick and place procedures'. In: *Proceedings of the 25th International Conference on Intelligent User Interfaces*. ACM, 2020. DOI: 10.1145/3377325.3377497.

[148] Philipp Kratzer et al. 'Anticipating Human Intention for Full-Body Motion Prediction in Object Grasping and Placing Tasks'. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 1157–1163. DOI: 10.1109/ro-man47096.2020.9223547.

[149] Richard G Freedman and Shlomo Zilberstein. 'Integration of planning with recognition for responsive interaction using classical planners'. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017. DOI: 10.1609/aaai.v31i1.11188.

[150] Peter Krauthausen and Uwe D Hanebeck. 'Intention recognition for partial-order plans using dynamic bayesian networks'. In: *2009 12th International Conference on Information Fusion*. IEEE. 2009, pp. 444–451.

[151] Dominik Riedelbauch, Daniel Luthardt-Bergmann and Dominik Henrich. 'A Cognitive Human Model for Virtual Commissioning of Dynamic Human-Robot Teams'. In: *2021 Fifth IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2021. DOI: 10.1109/irc52146.2021.00011.

[152] Yaqing Wang et al. 'Generalizing from a Few Examples: A Survey on Few-shot Learning'. In: *ACM Computing Surveys* 53.3 (2020), pp. 1–34. DOI: 10.1145/3386252.

[153] Alexander Gepperth and Barbara Hammer. 'Incremental learning algorithms and applications. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning'. In: *European symposium on artificial neural networks (ESANN)*. Ed. by Michel Verleysen. Louvain-la-Neuve, Belgique: Ciaco - i6doc.com, 2016.

[154] Yazan Abu Farha, Alexander Richard and Juergen Gall. 'When will you do what?-anticipating temporal occurrences of activities'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5343–5352. DOI: 10.1109/cvpr.2018.00560.

[155] Jianwu Fang et al. 'Behavioral Intention Prediction in Driving Scenes: A Survey'. In: *IEEE Transactions on Intelligent Transportation Systems* 25.8 (2024), pp. 8334–8355. DOI: 10.1109/tits.2024.3374342.

[156] Dominic A. Neu, Johannes Lahann and Peter Fettke. 'A systematic literature review on state-of-the-art deep learning methods for process prediction'. In: *Artificial Intelligence Review* 55.2 (2021), pp. 801–827. DOI: 10.1007/s10462-021-09960-8.

[157] Andrey Rudenko et al. 'Human motion trajectory prediction: a survey'. In: *The International Journal of Robotics Research* 39.8 (2020), pp. 895–935. DOI: 10.1177/0278364920917446.

[158] Ali Ghadirzadeh et al. 'Human-Centered Collaborative Robots With Deep Reinforcement Learning'. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 566–571. DOI: 10.1109/lra.2020.3047730.

[159] Jae Sung Park, Chonhyon Park and Dinesh Manocha. 'I-Planner: Intention-aware motion planning using learning-based human motion prediction'. In: *The International Journal of Robotics Research* 38.1 (2019), pp. 23–39. DOI: 10.1177/0278364918812981.

[160] Edoardo Alati et al. 'Help by Predicting What to Do'. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019. DOI: 10.1109/icip.2019.8803155.

[161] Martijn Cramer, Karel Kellens and Eric Demeester. 'Probabilistic Decision Model for Adaptive Task Planning in Human-Robot Collaborative Assembly Based on Designer and Operator Intents'. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7325–7332. DOI: 10.1109/lra.2021.3095513.

[162] Mesut Yang, Micah Carroll and Anca Dragan. *Optimal Behavior Prior: Data-Efficient Human Models for Improved Human-AI Collaboration*. 2022. DOI: 10.48550/ARXIV.2211.01602.

[163] Niek Tax et al. 'Predictive Business Process Monitoring with LSTM Neural Networks'. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2017, pp. 477–492. DOI: 10.1007/978-3-319-59536-8_30.

[164] Andrea Casalino et al. 'Predicting the human behaviour in human-robot co-assemblies: an approach based on suffix trees'. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020. DOI: 10.1109/iros45743.2020.9341301.

[165] Gyunam Park and Minseok Song. 'Predicting performances in business processes using deep neural networks'. In: *Decision Support Systems* 129 (2020), p. 113191. DOI: 10.1016/j.dss.2019.113191.

[166] Manuel Camargo, Marlon Dumas and Oscar González-Rojas. 'Learning Accurate LSTM Models of Business Processes'. In: *Business Process Management*. Springer International Publishing, 2019, pp. 286–302. DOI: 10.1007/978-3-030-26619-6_19.

[167] Asjad Khan et al. 'DeepProcess: Supporting Business Process Execution Using a MANN-Based Recommender System'. In: *Service-Oriented Computing*. Springer International Publishing, 2021, pp. 19–33. DOI: 10.1007/978-3-030-91431-8_2.

[168] Harish Chaandar Ravichandar et al. 'Learning and predicting sequential tasks using recurrent neural networks and multiple model filtering'. In: *2016 AAAI Fall Symposium Series*. 2016.

[169] Jianjing Zhang, Peng Wang and Robert X. Gao. 'Hybrid machine learning for human action recognition and prediction in assembly'. In: *Robotics and Computer-Integrated Manufacturing* 72 (2021), p. 102184. DOI: 10.1016/j.rcim.2021.102184.

[170] Joerg Evermann, Jana-Rebecca Rehse and Peter Fettke. 'Predicting process behaviour using deep learning'. In: *Decision Support Systems* 100 (2017), pp. 129–140. DOI: 10.1016/j.dss.2017.04.003.

[171] Abdulrhman Al-Jebrni, Hongming Cai and Lihong Jiang. 'Predicting the Next Process Event Using Convolutional Neural Networks'. In: *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*. IEEE, 2018. DOI: 10.1109/pic.2018.8706282.

[172] Julian Theis and Houshang Darabi. 'Decay Replay Mining to Predict Next Process Events'. In: *IEEE Access* 7 (2019), pp. 119787–119803. DOI: 10.1109/access.2019.2937085.

[173] Vincenzo Pasquadibisceglie et al. 'Using Convolutional Neural Networks for Predictive Process Analytics'. In: *2019 International Conference on Process Mining (ICPM)*. IEEE, 2019. DOI: 10.1109/icpm.2019.00028.

[174] A. J. Piergiovanni et al. 'Adversarial Generative Grammars for Human Activity Prediction'. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 507–523. DOI: 10.1007/978-3-030-58536-5_30.

[175] Chen Wang et al. 'Co-GAIL: Learning Diverse Strategies for Human-Robot Collaboration'. In: *Conference on Robot Learning*. PMLR. 2022, pp. 1279–1290.

[176] Yi Sun et al. 'Learn How to Assist Humans Through Human Teaching and Robot Learning in Human–Robot Collaborative Assembly'. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52.2 (2022), pp. 728–738. DOI: 10.1109/tsmc.2020.3005340.

[177] Miquel Ramírez and Hector Geffner. 'Goal recognition over POMDPs: Inferring the intention of a POMDP agent'. In: *IJCAI*. IJCAI/AAAI. 2011, pp. 2009–2014.

[178] Nakul Gopalan and Stefanie Tellex. 'Modeling and solving human-robot collaborative tasks using POMDPs'. In: *RSS Workshop on Model Learning for Human-Robot Communication*. 2015.

[179] Wolfgang Kratsch et al. 'Machine Learning in Business Process Monitoring: A Comparison of Deep Learning and Classical Approaches Used for Outcome Prediction'. In: *Business & Information Systems Engineering* 63.3 (2020), pp. 261–276. DOI: 10.1007/s12599-020-00645-0.

[180] Marcel Ph. Mayer et al. 'Cognitive Engineering of Automated Assembly Processes'. In: *Human Factors and Ergonomics in Manufacturing & Service Industries* 24.3 (2012), pp. 348–368. DOI: 10.1002/hfm.20390.

[181] Barbara Odenthal et al. 'Cognitive engineering for human-robot interaction: The effect of subassemblies on assembly strategies'. In: *Advances in Human Factors and Ergonomics Series*. CRC Press, 2010, pp. 200–209. DOI: 10.1201/ebk1439834992-22.

[182] Daniel Luthardt-Bergmann. 'Simulation-based user studies to observe human behavior for HRC experiments'. MA thesis. Germany: University of Bayreuth, 2021.

[183] Novie Susanto, Ratna Purwaningsih and Kharisma Panca Kurniawati. 'Design of the Human Assembly Strategy in a Self-Optimizing Assembly Cell: A Case Study of Indonesians'. In: *Makara Journal of Technology* 20.3 (2016), pp. 139–146. DOI: 10.7454/mst.v20i3.3069.

[184] Jeffrey Pennington, Richard Socher and Christopher Manning. 'Glove: Global Vectors for Word Representation'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014. DOI: 10.3115/v1/d14-1162.

[185] Heramb Nemlekar et al. 'Towards Transferring Human Preferences from Canonical to Actual Assembly Tasks'. In: *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 2022. DOI: 10.1109/roman53752.2022.9900872.

[186] Nelson Cowan. 'The magical number 4 in short-term memory: A reconsideration of mental storage capacity'. In: *Behavioral and Brain Sciences* 24.1 (2001), pp. 87–114. DOI: 10.1017/s0140525x01003922.

[187] Derya Soydaner. 'Attention mechanism in neural networks: where it comes and where it goes'. In: *Neural Computing and Applications* 34.16 (2022), pp. 13371–13385. DOI: 10.1007/s00521-022-07366-3.

[188] Wojciech Zaremba and Ilya Sutskever. *Learning to Execute*. 2014. DOI: 10.48550/ARXIV.1410.4615.

[189] Yangqing Jia et al. 'Caffe: Convolutional Architecture for Fast Feature Embedding'. In: MM '14 (2014). DOI: 10.1145/2647868.2654889.

[190] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. https://www.ruder.io/optimizing-gradient-descent. 2016.

[191] John Duchi, Elad Hazan and Yoram Singer. 'Adaptive Subgradient Methods for Online Learning and Stochastic Optimization'. In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159.

[192] Tengteng Zhang and Hongwei Mo. 'Reinforcement learning for robot research: A comprehensive review and open issues'. In: *International Journal of Advanced Robotic Systems* 18.3 (2021), pp. 1–22. DOI: 10.1177/17298814211007305.

[193] Florian Fischer. 'An Optimal Feedback Control Perspective on Human-Computer Interaction'. PhD thesis. 2024. DOI: 10.15495/EPUB_UBT_00007451.

[194] Richard Meyes et al. *Ablation Studies in Artificial Neural Networks*. 2019. DOI: 10.48550/ARXIV.1901.08644.

[195] Yanhao He and Steven Liu. 'Analytical Inverse Kinematics for Franka Emika Panda – a Geometrical Solver for 7-DOF Manipulators with Unconventional Design'. In: *2021 9th International Conference on Control, Mechatronics and Automation (ICCMA)*. IEEE, 2021. DOI: 10.1109/iccma54375.2021.9646185.

[196] Wietse van Dijk et al. 'The effect of human autonomy and robot work pace on perceived workload in human-robot collaborative assembly work'. In: *Frontiers in Robotics and AI* 10 (2023). DOI: 10.3389/frobt.2023.1244656.

[197] Sandra G. Hart. 'Nasa-Task Load Index (NASA-TLX): 20 Years Later'. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50.9 (2006), pp. 904–908. DOI: 10.1177/154193120605000909.

[198] Meia Chita-Tegmark et al. 'Can You Trust Your Trust Measure?' In: *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2021. DOI: 10.1145/3434073.3444677.

[199] S.M. Mizanoor Rahman and Yue Wang. 'Mutual trust-based subtask allocation for human–robot collaboration in flexible lightweight assembly in manufacturing'. In: *Mechatronics* 54 (2018), pp. 94–109. DOI: 10.1016/j.mechatronics.2018.07.007.

[200] Jonathan Hümmer, Dominik Riedelbauch and Dominik Henrich. 'Comparing the Consistency of User Studies Conducted in Simulations and Laboratory Settings'. In: *2024 Eighth IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2024, pp. 154–161. DOI: 10.1109/irc63610.2024.00034.

[201] Tommaso Panchetti et al. 'Assessing the Relationship between Cognitive Workload, Workstation Design, User Acceptance and Trust in Collaborative Robots'. In: *Applied Sciences* 13.3 (2023), p. 1720. DOI: 10.3390/app13031720.

[202] Costanza Messeri et al. 'Human-robot collaboration: optimizing stress and productivity based on game theory'. In: *IEEE Robotics and Automation Letters* (2021), pp. 1–1. DOI: 10.1109/lra.2021.3102309.

[203] Piotr Fratczak. 'Trust and Fluency in Industrial Human-robot Interaction: Virtual Reality Study of Human Behaviour'. PhD thesis. Loughborough University, 2020.

[204] John D Lee and Katrina A See. 'Trust in automation: Designing for appropriate reliance'. In: *Human factors* 46.1 (2004), pp. 50–80. DOI: 10.1518/hfes.46.1.50.30392.

[205] Aya Hussein. 'Performance, Trust, and Transparency for Effective Human-Swarm Interaction'. PhD thesis. UNSW Sydney, 2021.

[206] William S. Helton, Gregory J. Funke and Benjamin A. Knott. 'Measuring workload in collaborative contexts: Trait versus state perspectives'. In: *Human Factors* 56.2 (2014), pp. 322–332. DOI: 10.1177/0018720813490727.

[207] Mehrnoosh Askarpour et al. 'Formal model of human erroneous behavior for safety analysis in collaborative robotics'. In: *Robotics and Computer-Integrated Manufacturing* 57 (2019), pp. 465–476. DOI: 10.1016/j.rcim.2019.01.001.

[208] Fabio Bonsignorio. 'A New Kind of Article for Reproducible Research in Intelligent Robotics [From the Field]'. In: *IEEE Robotics and Automation Magazine* 24.3 (2017), pp. 178–182. DOI: 10.1109/mra.2017.2722918.

[209] IEEE. *Reproducible Articles (R-Articles), Short Replication Articles (r-articles), Reply Articles*. https://ieee-ras.org/publications/ram/information-for-authors-ram/reproducible-articles-r-articles-short-replication-articles-r-articles-reply-articles. [Online; accessed 2024-06-11]. 2017.

[210] Peter Corke and Jesse Haviland. 'Not your grandmother's toolbox–the Robotics Toolbox reinvented for Python'. In: *IEEE International Conference on Robotics and Automation*. IEEE, 2021. DOI: 10.1109/icra48506.2021.9561366.

[211] Vidyasagar Rajendran et al. 'A Framework for Human-Robot Interaction User Studies'. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020. DOI: 10.1109/iros45743.2020.9341286.

[214] Goolge Developers. *Overview | Procotocal Buffers Documentation*. https://web.archive.org/web/20230123205854/https://protobuf.dev/overview/. [Online, accessed 2024-02-06]. 2023.

[215] Martin A. Fischler and Robert C. Bolles. 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography'. In: *Communications of the ACM* 24.6 (1981), pp. 381–395. DOI: 10.1145/358669.358692.

[216] Oliver Siegried Zahn. *Integration der HoloLens 2 in FlexCobot*. BA thesis. Germany, 2021.

[217] Jiaolong Yang et al. 'Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.11 (2016), pp. 2241–2254. DOI: 10.1109/tpami.2015.2513405.

[218] Microsoft Corporation. *Spatial Anchors*. https://web.archive.org/web/20221108151420/https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-anchors. [Online; accessed 2024-06-02]. 2022.

[219] Keith Vertanen. *NASA Task Load Index*. https://www.keithv.com/software/nasatlx/nasatlx.html. [Online; accessed 2024-03-1]. 2011.

Prior publications of the author (peer-reviewed)

[19] Dominik Riedelbauch, Nico Höllerich and Dominik Henrich. 'Benchmarking Teamwork of Humans and Cobots – An Overview of Metrics, Strategies, and Tasks'. In: *IEEE Access* 11 (2023), pp. 43648–43674. DOI: 10.1109/access.2023.3271602.

[69] Nico Höllerich and Dominik Henrich. 'Relevant Perception Modalities for Flexible Human-Robot Teams'. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020. DOI: 10.1109/roman47096.2020.9223593.

[104] Nico Höllerich and Dominik Henrich. 'Coloured Petri Nets for Monitoring Human Actions in Flexible Human-Robot Teams'. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021. DOI: 10.1109/iros51168.2021.9636428.

[220] Johannes Doleschal et al. 'Chisel'. In: *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*. ACM Press, 2018. DOI: 10.1145/3184558.3186963.

Prior publications of the author (other)

[212] Nico Höllerich. *Fluency in Dynamic Human-Robot Teaming with Intention Prediction - Main Application*. 2025. DOI: 10.5281/ZENODO.15783114.

[213] Nico Höllerich. *Fluency in Dynamic Human-Robot Teaming with Intention Prediction - Data and Plots*. 2025. DOI: 10.24433/CO.2820731.V1.

[221] Nico Höllerich. *Reflexionsberichte von ZHL-Lehrwerkstatt Teilnehmenden aus dem Wintersemester 2022/23*. Ed. by Anja Hager. Bayreuth, 2023.

[222] Nico Höllerich and Dominik Henrich. 'AR verbindet Mensch und Roboter'. In: *Industrial Production* (2022), pp. 46–47.

[223] Michael Gradmann et al. 'Handbuch Mensch-Roboter-Kollaboration'. In: ed. by Rainer Müller et al. 2., aktualisierte Auflage. München: Hanser, 2024. Chap. Unimodale Ineraktion unter Nutzung visueller Schnittstellen, pp. 181 –183. DOI: 10.1007/978-3-446-47460-4.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet habe. Weiterhin erkläre ich, dass ich die Hilfe von gewerblichen Promotionsberatern bzw. –vermittlern oder ähnlichen Dienstleistern weder bisher in Anspruch genommen habe, noch künftig in Anspruch nehmen werde.
Zusätzlich erkläre ich hiermit, dass ich keinerlei frühere Promotionsversuche unternommen habe.

Bayreuth, den 4. Juli 2025

Nico Höllerich