Diss. ETH No. 26759

Interacting with Smart Devices – Advancements in Gesture Recognition and Augmented Reality

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

Vincent Georg Ernest Becker
M.Sc. in Informatics, Karlsruhe Institute of Technology
born on 17th December 1991
citizen of Germany and
the United Kingdom of Great Britain and Northern Ireland

accepted on the recommendation of
Prof. Dr. Friedemann Mattern, examiner
Prof. Dr. Otmar Hilliges, co-examiner
Prof. Dr. Kai Kunze, co-examiner

2020

Dedicated to my parents Louise and Bernd-Dietmar

Abstract

Over the last decades, embedded computer systems have become powerful and widespread with remarkable success. Besides traditional computers, such as desktops, laptops, smartphones, and servers, such systems have become part of nearly every technical appliance, for example, cars, televisions, and washing machines and thereby, an essential part of our lives. A common phrase for such appliances is "smart device", a term which encompasses equipment to which one can digitally connect in order to exchange information and commands. Further, it can potentially sense its environment and process as well as act upon this measurement.

Although computers and the devices containing them play such an important role, the way in which we interact with them has not changed much since the early days. Humans are required to control them in a manner very different from human-to-human communication. In particular, the possibilities to provide input to a smart device are mostly limited to traditional interfaces, such as buttons, knobs, or keyboards; or graphical representations thereof on displays.

Technological progress in recent years in hardware, as well as in algorithmic methods, e.g. in machine learning, enables novel solutions for human-computer (or in fact human-device) interaction. Only recently, the use of speech has become practical and is used on smartphones as well as for home devices, incorporating a modality in the interaction process that is innate to humans and rich in expressiveness. However, for natural communication, other modalities, such as gestures, complement speech and may do so in human-computer communication, enabling simple and spontaneous interactions and avoiding the known social awkwardness of having to talk to devices. The adoption of speech and also other commonly used interaction methods, such as touch input on smartphones, indicate the relevance of considering further modalities in addition to the traditional ones. This dissertation contributes to further bridging the interaction gap between humans and smart devices by exploring solutions in the following areas:

(i) Wearable gesture recognition based on electromyography (EMG): The touch of our fingers is widely used for interaction. However, most approaches only consider binary touch events. We present a method, which classifies finger touches using a novel neural network architecture and estimates their force based on data recorded from a wireless EMG armband. Our method runs in real time on a smartphone and allows for new interactions

with devices and objects, as any surface can be turned into an interactive surface and additional functionality can be encoded through single fingers and the force applied.

(ii) Wearable gesture recognition based on sound and motion: Besides other signals, gestures might also emit sound. We develop a recognition method for sound-emitting gestures, such as snapping, knocking, or clapping, employing only a standard smartwatch. Besides the motion information from the built-in accelerometer and gyroscope, we exploit audio data recorded by the smartwatch microphone as input. We propose a lightweight convolutional neural network architecture for gesture recognition, specifically designed to run locally on resource-constrained devices. It achieves a user-independent recognition accuracy of 97.2% for nine distinct gestures. We find that the audio input drastically reduces the false positive rate in continuous recognition compared to using only motion.

(iii) Device representations in wearable augmented reality (AR): While AR technology is becoming increasingly available to the public, ways of interaction in the AR space are not yet fully understood. We investigate how users can control smart devices in AR. Connected devices are augmented with interaction widgets representing them. For example, a widget can be overlaid on a loudspeaker to control its volume. We explore three ways of manipulating the virtual widgets in a user study: (1) in-air finger pinching and sliding, (2) whole-arm gestures rotating and waving, (3) incorporating physical objects in the surroundings and mapping their movements to interaction primitives. We find significant differences in the preference of the users, the speed of executing commands, and the granularity of the type of control. While these methods only apply to control of a single device at a time, in a second step, we create a method which also takes potential connections between devices into account. Users can view, create, and manipulate connections between smart devices in AR using simple gestures.

(iv) Personalizable user interfaces from simple materials: User interfaces rarely adapt to specific user preferences or the task at hand. We present a method that allows the quick and inexpensive creation of personalized interfaces from paper. Users can cut out shapes and assign control functions to these paper snippets via a simple configuration interface. After configuration, control takes place entirely through the manipulation of the paper shapes, providing the experience of a tailored tangible user interface. The shapes, which are monitored by a camera with depth sensing, can be dynamically changed during use.

The proposed methods aim at a more natural interaction with smart devices through advanced sensing and processing in the user's environment or on his/her body itself. As these interactions could be made ubiquitously available through wearable computers, our methods could help to improve the usability of the growing number of smart devices and make them more easily accessible to more people.

Kurzfassung

In den letzten Jahrzehnten sind eingebettete Computersysteme sehr leistungsfähig geworden und haben eine weite Verbreitung gefunden. Neben dem Einsatz in herkömmlichen Datenverarbeitungs- und Kommunikationssystemen für Konsumenten, wie Desktops oder Mobiltelefonen, finden Computer heutzutage in beinahe jedem technischen Gerät Verwendung, beispielsweise in Autos, Unterhaltungs- und Haushaltsgeräten. Computer sind dadurch zu einem festen Bestandteil unseres Alltags geworden. Eine geläufige Bezeichnung für solche Geräte mit eingebauten Computern ist "Smart Device". Dieser Begriff umfasst Apparate, mit denen man eine digitale Kommunikationsverbindung aufbauen kann, um Daten austauschen und Kommandos absetzen zu können. Ein solcher Apparat kann eventuell auch mittels eingebauter Sensoren relevante Zustandsinformationen messen und aufgrund dieser Messung agieren.

Obwohl Computer und die Geräte, in welche diese verbaut sind, eine zunehmend wichtigere Rolle spielen, hat sich die Art und Weise, wie wir mit ihnen interagieren, seit Anbeginn ihrer Verwendung nur wenig geändert. Die Kommunikation zwischen Mensch und Maschine findet auf eine Weise statt, die sich deutlich von zwischenmenschlicher Kommunikation unterscheidet. Insbesondere sind die Möglichkeiten, Eingaben zu tätigen, meist auf herkömmliche Schnittstellen wie mechanische Elemente, z.B. Knöpfe oder Tastaturen, eingeschränkt, bzw. auf eine graphische Darstellung davon.

Der technische Fortschritt der letzten Jahre, sowohl im Hardwarebereich als auch bei algorithmischen Methoden, beispielsweise im Bereich des maschinellen Lernens, erlaubt neue Arten der Mensch-Maschine-Kommunikation. Sprachsteuerung beispielsweise, ermöglicht durch automatische Spracherkennung, ist mittlerweile für viele Mobiltelefone und verschiedene Heimsteuerungssysteme verfügbar. Sprache ist eine dem Menschen angeborene Kommunikationsfähigkeit, die vielfältige Ausdrucksmöglichkeiten und eine hohe Informationsdichte erlaubt. Jedoch gibt es neben gesprochener Sprache weitere Modalitäten der menschlichen Kommunikation, die diese ergänzen, wie z.B. Gesten. Diese können auch in der Mensch-Maschine-Kommunikation genutzt werden, um schnelle und einfache Interaktionen zu ermöglichen. Die Verwendung von Spracherkennung, aber auch von anderen Technologien wie berührungssensitiven Bildschirmen, deutet auf die Relevanz der Verwendung weiterer Modalitäten zur Interaktion hin. Die vorliegende Dissertation soll dazu beitragen, die Interaktionsmöglichkeiten zwischen Menschen und "Smart Devices"

in diesem Sinne zu erweitern, insbesondere durch Lösungen in den folgenden Bereichen:

(i) Gestenerkennung mittels mobiler Elektromyographie (EMG): Viele Interaktionen mit Geräten beruhen auf Berührungen mit den Fingern, sei es bei mechanischen Schnittstellen oder beispielsweise berührungssensitiven Bildschirmen. Die meisten Methoden zur Erkennung dieser Berührungen erfassen diese jedoch nur als binäre Ereignisse. Wir stellen eine Methode vor, die unter Verwendung eines neuronalen Netzes bestimmt, mit welchem Finger eine Berührung durchgeführt wird und wie viel Kraft durch den Finger ausgeübt wird. Diese Klassifizierung und Kraftschätzung basiert auf Messungen eines kabellosen EMG-Armbands. Unser Algorithmus läuft dabei in Echtzeit auf einem handelsüblichen Mobiltelefon und ermöglicht neue Interaktionsmöglichkeiten mit Geräten und Objekten, da jede Oberfläche ohne Notwendigkeit einer Veränderung zur Interaktion genutzt werden kann und zusätzliche Funktionen durch die einzelnen Finger und die ausgeübte Kraft kodiert werden können.

(ii) Gestenerkennung basierend auf Audio- und Bewegungsdaten mobiler Sensorik: Verschiedene Handgesten erzeugen neben reinen Bewegungen auch Geräusche, wie z.B. Klatschen, Schnipsen oder Klopfen. Wir entwickeln eine Methode, die diese Geräusche zur Erkennung der Gesten nutzt. Dazu verwenden wir eine handelsübliche Smartwatch, die neben Bewegungsdaten mittels eines eingebauten Intertialsensors auch Audiodaten mittels eines Mikrofons aufnimmt. Die Daten werden direkt auf der Smartwatch in Echtzeit durch ein neuronales Netz klassifiziert. Dieses erreicht eine nutzerunabhängige Erkennungsgenauigkeit von 97.2% für neun Gesten. Dabei zeigt sich, dass die Audiodaten zu einer starken Reduzierung der Falschpositivrate beitragen.

(iii) Repräsentation von Geräteschnittstellen durch Augmented Reality (AR): AR-Technologie wird durch die Entwicklung und den Vertrieb entsprechender Brillen in immer weiteren Kreisen verfügbar. Dies eröffnet völlig neue Interaktionsmöglichkeiten und es ist noch unklar, wie die Interaktion in AR am besten stattfinden soll. Wir untersuchen, wie Benutzer mittels AR mit “Smart Devices” interagieren können, indem wir deren Benutzerschnittstellen in AR repräsentieren und verschiedene Interaktionsmöglichkeiten in einer Nutzerstudie vergleichen: Eingaben durch (1) Gesten, mit denen man die visualisierten Schnittstellenelemente virtuell greifen und verschieben kann, (2) rotieren des Arms und Wischgesten, (3) Einbeziehung von physischen Objekten in der Umgebung, die bewegt werden können, wobei die Bewegungen auf Veränderungen der virtuellen Elemente abgebildet werden. Dabei finden wir signifikante Unterschiede in den Präferenzen der Nutzer und der Geschwindigkeit und der Granularität der verschiedenen Methoden. Da diese Methoden nur die Bedienung und Konfiguration eines einzelnen Geräts erlauben, erweitern wir unser Konzept mit der Möglichkeit, in AR mittels Gesten Verbindungen zwischen verschiedenen Geräten zu erzeugen, die dann direkt in der Systemkonfiguration abgebildet werden.

(iv) Personalisierbare Benutzerschnittstellen, erzeugbar aus einfachen Materialien: Die meisten Benutzerschnittstellen passen sich weder an den Benutzer noch an die gegenwärtige Aufgabe an. Wir stellen eine Methode vor, mit welcher der Benutzer selbst aus einfachen und kostengünstigen Materialien in kurzer Zeit eine Benutzerschnittstelle erstellen kann. Die gewünschten Interaktionselemente können frei aus Papier ausgeschnitten werden und zur erdachten Schnittstelle zusammengestellt werden. Daraufhin kann der Nutzer den einzelnen Papierformen Funktionen zuweisen. Die Formen, sowie die Finger des Benutzers, werden durch eine Farb- und Tiefenkamera verfolgt, wodurch die Bewegungen von unserer Methode wahrgenommen werden. Dadurch kann der Nutzer nach der Konfiguration alle definierten Interaktionen auch tatsächlich durchführen. Neben den vielfältigen Gestaltungsmöglichkeiten hat unsere Methode gegenüber herkömmlichen graphischen Benutzerschnittstellen den Vorteil, dass sie dem Nutzer einen haptischen Eindruck vermittelt.

Die vorgestellten Methoden zielen auf eine natürlichere Interaktion mit “Smart Devices” ab, ermöglicht durch neuartige Sensorik mit Informationsverarbeitung in der Umgebung des Nutzers oder in tragbarer Form beim Nutzer selbst. Da auf diese Weise die Interaktionen durch mobile, eingebettete Computer allgegenwärtig verfügbar gemacht werden können, können unsere Methoden dazu beitragen, die Interaktionen mit der stetig wachsenden Anzahl von “Smart Devices” einfacher zu gestalten, die Benutzbarkeit dieser Geräte zu verbessern und sie für eine grössere Zielgruppe zugänglich zu machen.

Acknowledgements

During my doctoral studies at ETH, I received support from many people, without whom this work would not have been possible. First of all, I would like to thank my supervisor, Prof. Friedemann Mattern, for letting me join his research group and thereby making this whole project possible, for sharing his vision and backing my work over the last years, and for providing an accommodating environment at ETH for all group members. Furthermore, I want to thank my co-supervisors Prof. Otmar Hilliges and Prof. Kai Kunze for their efforts in supporting this dissertation.

With great gratitude and nice memories, I would like to thank my colleagues and friends from all over the world that I met at the Distributed Systems Group. Of course, the set of people in the group changed over the years. From those, who have left by now, I want to thank Wilhelm Kleiminger, for making sure I settled in well and for introducing me to my first research topic in the field of smart energy, Leyna Sadamori for interesting discussions and clear opinions, and Anwar Hithnawi and Hossein Shafagh for being our hard-working and successful PhD student role models. In particular, I would like to thank Gábor Sörös, who shared his fascination on research with us, directly supported most of my projects and helped with several publications. Further thanks go to Hông-Ân Cao, Marian George and Matthias Kovatsch.

From the people currently present in the group, I first want to thank Mihai Bâce, who has been a great office neighbour over the last years, always considerate, supportive, and humorous. I would also like to thank Lukas Burkhalter and Alexander Viand for helping with and eventually taking over the organisation of our Informatics II lecture. Furthermore, I would like to extend my thanks to Jing Yang and Liliana Barrios for being so pleasant and happy colleagues and thereby brightening our office environment. Thanks to Vlad Coroamă for always being in a jovial mood and for his support in the final smart energy project. Thanks to Simon Mayer for exciting discussions, a project collaboration, and never-ending optimism. Last, but certainly not least, I want to thank our secretary Barbara

von Allmen-Wilson, who was always very helpful and did her best to support us. I hope that we will all keep in touch and meet many times again in the future.

During my time as a PhD student, I had the pleasure of supervising several bright-minded students at ETH, whom I would like to thank for their hard work and collaboration in their projects: Erfan Abdi, Linus Fessler, Sandro Kalbermatter, Alexander Kayed, Christian Mnuk, Pietro Oldrati, Felix Rauchenstein, Chenyang Wang, and Maximilian Wurm. I wish them all the best for their personal and professional future.

For the longest support, I surely owe the most gratitude to my family, my siblings Philipp, Marie-Claire, and Oliver, and especially my parents Louise and Bernd-Dietmar, who inspired me for science and engineering from an early age on. They did everything they could to support me in every possible way.

Finally, I would like to thank Selina for her never-ending support and confidence over the last years, in times of success, but also in times of frustration. I will be forever grateful.

Contents

Chapter 1

Introduction

In 1991, Marc Weiser published his article “The Computer for the 21st Century” [180], coining the term “ubiquitous computing”. Therein, he propagates his vision of a system of numerous computers incorporated in our environment or carried on our bodies, interconnected and working together to provide us with advanced services and utility in our daily lives, rather than humans using the then common single powerful computer. With their great success in recent decades, computers have indeed spread into all appliance domains, often in the form of embedded systems. This dissemination brings us closer to Marc Weiser’s vision of ubiquitous computing, in which many distributed computers in our environment enhance our daily lives and help us to fulfil our tasks. Furthermore, products become easier to produce and update, as control functions can be implemented in software.

Embedded computers and communication technology have on the one hand turned ordinary appliances in our daily lives such as washing machines or cars into “smart devices”, and on the other, we use new devices which enable continuous and fast access to and exchange of information. The rapid dissemination of smartphones and their adoption as the most personal devices [220], which almost everyone carries with themselves at nearly all times, are an evident example of these developments. A smart device is a device containing a computer system, to which we can establish a digital communication connection, which to some extent enables an interactive operation during which information about the device’s state may be queried or commands transmitted to the device. Often, this connection is provided by wireless communication. For example, a smart light can usually be controlled via a smartphone and enables the changing of the brightness or colour in a smartphone app. That means that a smart device does not have to have a user interface of its own. The interface “may vanish altogether” and can be “exported” to

another device [139]. Smart devices may contain sensors to measure the state of their surroundings and may be able to process and act upon this measured information.

Although people like Weiser surely thought about computers being embedded in everyday objects to enable the concept of ubiquitous computing [181], the term "smart device" or synonyms such as "smart object", "smart thing", and "smart appliance" were probably first used only several years later around the turn of the millennium [2, 3, 8, 36, 92, 107, 122, 139, 179]. Precursors to the term were expressions such as "computer-augmented environments" [184], "informative things" [18], "intelligent device" [45], "proactive systems" [164], and "things that think" [145].

While the term "smart" suggests an underlying concept of intelligence, smart devices do not necessarily have to incorporate a form of artificial intelligence. It is instead the idea that we can communicate with them, which lets them appear smart to us. A temperature sensor with a Bluetooth module through which a smartphone app can query the current temperature can be seen as a smart device without exhibiting any feature of intelligence. Nevertheless, smart devices may provide advanced services to their users, for example, a smart meter, an electricity meter which can be queried directly via the internet, can inform the inhabitants about their current energy consumption [182]. Furthermore, it might even estimate the occupancy state based on that consumption and control the heating to save energy in times of absence [25, 26]. A car might immediately recognize, who is in the driver's seat and adjust the seat and driving settings. Besides these passive services, we can also actively use our smart devices, such as smartphones to obtain access to information at virtually any place and any time. A particular class of smart devices are wearables, such as smartwatches or smart glasses. These can enhance the human's abilities, as they can sense the world from an egocentric perspective, thereby naturally perceiving what the human perceives: cameras can see what the human sees, the human's location and motion can be measured, body sensors can potentially sense the mental and emotional state.

Besides devices which contain computers themselves, ordinary objects can also be made "smart" by adding communication abilities to them [122]. These are then referred to as smart objects. This "upgrade" can, for example, be achieved through RFID (radio-frequency identification) tags or visual markers such as barcodes, which can be read using another device such as a smartphone. The tags or codes convey information about the object either directly or provide a link to a web resource with more information. While the object itself may not incorporate any information processing capability itself, it still appears smart to the user because of the possibility to interact and exchange information. One might scan a barcode of a food item with a shopping app on one's smartphone, retrieving the nutrition and allergen information. Exploiting one's preferences stored in the app, e.g. about allergies to certain substances, it would then seem like the food item itself was telling us whether it was fit for consumption. As such tags or markers are inexpensive,

this concept allows us to make virtually any object smart. This implies that the number of smart device instances which participate in the ubiquitous network of devices and objects, also referred to as the "Internet of Things", and the number of devices that we might be interacting with in the future, could be much larger than the mere number of devices containing actual information processing units.

Besides the mere dissemination of computers in our environment and the added utility for their users through the provided services, Weiser also conveyed another, likely more important message, which he states right in the beginning in the first two sentences of his essay: "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." The computing technologies, which can be added or integrated to virtually any object as mentioned above, should not be apparent to the user, they should rather be invisible in a sense that a user does not realize he or she is actually interacting with a computer at all. The interaction should be human-centred, not focusing on the interaction with a computer, i.e. "the challenge of reducing the need for explicit human-computer interaction (HCI) becomes central." [55].

While computers have often changed the nature of appliances completely, have mostly replaced mechanical control systems, or have made the creation of certain systems possible at all, reducing their price by orders of magnitude, the way we interact with these appliances is still far from the notion of a seamless and natural, human-centred communication. People still use buttons and knobs to specify their input, either on the devices themselves or through remote controls. For traditional computers, we use keyboards for single character input and mice for simple pointing and clicking. With newer touch-based displays, we still control the devices through the use of virtual keyboards, buttons, sliders, and knobs.

The human often has to adapt to the device specification, making mistakes the human's fault, whereas, on the contrary, the computer should adapt to the human's needs. Appliances, or the set of appliances in our environment altogether, should provide a multi-modal, natural, robust, and fault-tolerant way of interaction. They should adapt to the human's context, and in the future, allow closer collaboration between humans and machines.

Making human-computer interaction more similar to human-human interaction, so that the human may not even notice that there is a computer, is a very challenging problem because the computer system has to infer the human's intention. The intention might be expressed through a combination of multiple modalities, e.g. speech, gestures, eye gaze, body pose, and emotional expressions, and depends on the human's context. The human brain undergoes a lot of learning in the area of observing and inferring the intention of other humans and in the end is highly trained and skilled in this process [10, 34, 165]. This ability not only facilitates interaction but was also essential for survival in the far past. And it might still be nowadays, e.g. when a car driver realizes that a pedestrian intends to

cross the street through an analysis of the pedestrian's movements, facial expression, and an anticipation of the pedestrian's expectations towards the driver him- or herself, which also links to the social and cultural context. This example is a complex task which still constitutes a great challenge for autonomous vehicles.

In this dissertation, we investigate different methods to enable a more seamless process of interacting with devices through advanced sensing and processing on and around the user. In particular, we focus on a sub-area of the challenges mentioned above: how a human can actively provide input to a smart device, for example, to issue commands.

In recent years, many device manufacturers have been providing smartphone apps to control the devices, which also allows making the device's physical interface as simple as possible. For example, music players often have a very simplistic physical user interface and are mainly controlled by apps, running on a smartphone, i.e. the device externalizes its interface to the app. This idea of "beaming" a device's interface to a smartphone has also been explored in research [124]. Using a smartphone as a universal control is a reasonable approach, as it provides a flexible framework for the design of graphical user interfaces, moreover it is inexpensive and robust. However, being a remote control, the smartphone constitutes an intermediary in the interaction of the human with the device. Instead of interacting with the device directly, the human now interacts with his/her smartphone.

One of the most significant advances in the field of natural human-computer communication has recently been made in the area of speech recognition and also dialogue modelling. Speech is one of the primary means of communication in human-to-human communication and has several advantages over other modes of communication: it can encode very complex content, and it provides a high information throughput. Therefore, it is undoubtedly desirable to enrich the space of interaction by speech recognition and synthesis. Several commercial products based on speech recognition have emerged in recent years, first on smartphones with Apple's Siri [208] and Google's assistant [223], then home control systems, such as Amazon's Alexa [206] and Google Home [224]. However, pure speech interaction systems have several shortcomings. First of all, people feel a social awkwardness of speaking to no one in particular when controlling a device through speech, epitomized by the phrase of feeling like "talking to a wall" [52]. When being together with others, speaking with a device also appears unnatural. Besides, it is prone to environmental noise. Most importantly, human communication is much richer and comprises many more modalities than only speech. We also use gestures and implicitly infer a lot of knowledge from the context we are in, both our internal context and that of our counterpart, and the environmental context we are placed in. If speech is used as the only modality, the user has to encode all the information usually conveyed through the other modalities. For example, instead of being able to point at a device which should be turned on, the user would have to select the device through spoken text, which is unnatural

and entails longer speech sequences than usually required.

We investigate several modalities for providing input to smart devices other than speech. We begin by examining different methods for gestural input, for example how to sense touch input on virtually every surface. Similarly to what the tags and markers do for the smart objects, this turns any surface into an interactive surface. Besides merely adding sensing and processing to recognize actions in the environment, we can also use augmented reality (AR) technology to change the way we perceive the world. We can use AR to display device representations in the user's first-person view on the real devices, thereby allowing for a form of direct interaction as the representations are collocated with the device. Finally, we create a system, which allows the user to create his/her own functioning interfaces from simple materials such as paper. Our approach enables improved interface prototyping but might also lead to a new form of personalized interfaces, created on demand by the users themselves in the future.

In all proposed methods in this thesis apart from the last one, we focus on making the developed methods available through wearable computers (the last interaction method could potentially be implemented using wearables). This has become feasible through the spread of wearables with increased sensing and computation capabilities, such as smartphones, smartwatches, fitness trackers, smart glasses, AR headsets, and many more. Augmenting the human has the great advantage of taking an egocentric perspective. Additionally, these wearables can recognize input from the human to control the device. Finally, the devices themselves are not required to incorporate any sensing and intelligence themselves, as the knowledge is acquired by the egocentric system directly. All that remains on the device side is to offer an application programming interface (API) to receive commands. Hence, the device can become as simple as possible. Examples pointing in this direction are devices which export their user interfaces to smartphones apps, such as music streamers controlled by an app as mentioned before or rental bicycles for which the payment process is handled via the smartphone.

In the rest of this chapter, we first give a brief overview of methods and modalities for providing input in human-computer interaction, before describing the contributions of this thesis in more detail.

1.1 Overview of Interaction Modalities for Providing Input

Human-computer interaction is a vast research field as such. It interweaves user-centred research on user experience and user-centred design with technical research efforts in

which the way to make interactions technically possible and improve them is investigated, e.g. which and how specific modalities can be used for interaction. As the main contributions in this dissertation are part of the latter area, we give a brief overview of the different modalities that have been employed for human-computer interaction, in particular concerning the issue of how the human can provide input. First of all, we want to give a short overview of the history of user interfaces as an introduction to the topic.

1.1.1 A Very Short History of User Interfaces

User interfaces came into existence long before the invention of computers. As soon as humankind invented tools and machinery, it was also necessary to be able to trigger actions, which naturally took place through mechanical controls, such as levers or knobs. Over time, the complexity of the interface increased with the complexity of the machines because of the necessity for setting many system parameters. For example, as Figure 1.1a shows, the controls of a steam train consisted of several knobs and levers. Interestingly, modern train interfaces are not dissimilar in terms of their elements and complexity, as Figure 1.1b shows. Many of these interfaces prevailed until today in household appliances, such as ovens or radios.

Later, with the emergence of electrical systems, control interfaces became necessary as the user could not interact with the electric current directly. Buttons and switches, e.g. for turning devices on and off, or levers and knobs for setting continuous values, evolved as the most common interfaces. When first communication and text processing systems, such

(a) Taken from [246].

(b) Taken from [250].

Figure 1.1: The user interfaces of a steam train and a modern train.

as typewriters, were developed, electromechanical keyboards were introduced to achieve higher throughput. The first patent for a typewriter was issued as early as 1714 in the UK. Keyboards were then also later adopted for computers, as shown in Figure 1.2. They still are one of the primary input devices for computer systems, also for touch devices, such as smartphones, where the keyboard is graphically represented, and the optimization of the layouts are an active area of research [89].

The emergence of computers changed the interaction paradigm for many tasks. All the attention is focused on a single universal machine, which is in contrast to the usual usage patterns, where every device has its dedicated function, and there are many devices in use. As a consequence, the computer required a relatively large amount of configuration or instructions to fulfil a specific task, which the users innately are not used to, especially since early versions were purely text-based. In the 1960s and 1970s several improvements to the interaction with a computer were made, e.g. the development of graphical user interfaces and computer mice (cf. Figure 1.3). These interfaces took advantage of the strong human capabilities in visual perception and fine-grained movements of the hand and also afforded a simpler mental model of the interaction with a computer. Around the same time, first touchscreens were developed, which enabled the direct interaction with the hand and fingers. While these methods were mostly used for actual computers, most other technical devices were controlled with ordinary buttons or switches, potentially outsourced to remote controls, which were introduced in the late 1930s (at first for radios). As envisioned and promoted by Marc Weiser's group at Xerox PARC (cf. Figure 1.4), the number, type, and use cases of computer devices have transitioned from one single central computer to several, which also includes tablets and smartphones, which we can

Figure 1.2: An Apple II computer with an integrated keyboard (presented in 1977). Taken from [240].

(a) A prototype of a computer mouse built by William English according to a design by Douglas Engelbart. Taken from [249].

(b) A computer mouse developed by the German company AEG Telefunken. Taken from [245].

Figure 1.3: Two computer mice independently developed and presented in 1968.

carry with us. In fact, smartphones have become the primary personal computing device at least for young people [220], used as a mobile personal assistant and access point to many services. In 2017 and 2018, mobile phones were responsible for around 50% of website traffic [48].

Figure 1.4: Computer devices with different form factors and purposes used for work at the Xerox PARC laboratory. Taken from [180].

While other interaction techniques, such as speech, may have been under development for a considerable amount of time, many of them have only recently become practically usable through the advances in algorithmic methods in machine learning, hardware, and higher communication bandwidths. Improvements in machine learning enable better signal processing and recognition techniques. Advances in hardware do not only provide more computing power, but also cheaper, smaller, and mobile devices, which we can carry with us and that can sense their environment. Enhanced communication allows devices to send data to more powerful servers for processing (which is becoming less important, as the devices "on the edge" are becoming more computationally powerful themselves). The following sections should give a brief overview of the main modalities which are being used or are under research for providing input to computer systems today.

1.1.2 Speech

Speech is a skill innate to humans and one of the essential communication mechanisms in human-to-human interaction. The processing of spoken language is a task the human brain is highly trained to perform [84]. Interestingly, speaking and processing speech is a distinctively human ability and is by far not as strongly developed even in the human's closest relatives, monkeys. Speech allows for a very high information throughput, which, besides the innateness to humans, makes it a valuable modality for human-computer interaction. Other advantages are that speech does not require the user to look at a specific interface or employ his/her hands, which is important when performing activities such as driving a car. However, due to the complexity of the speech signal, but also of the meaning of the words, automatic speech recognition and the subsequent processing of the natural language, i.e. the semantic understanding of the spoken information, is very challenging. Research on automatic speech recognition began in the 1950s, yet with limited success. It was only possible to recognize a few single words under lab conditions. First commercial systems for dictating and automatically transcribing spoken text were presented in the 1990s [226]. However, the recognition usually had to be adapted to each speaker, and there was no semantic processing of the text. A speech recognition system usually consists of an acoustic model, which is used to analyse the actual speech signal, and a language model, which can compute the probability of hypothesized phrases. Traditionally both of these parts were statistical models, such as Hidden Markov Models used as acoustic models [57, 88, 117]. Later, these tasks have mostly been taken over by neural networks, which led to significant performance improvements. Especially modelling the temporal structure of speech in neural networks proved to be beneficial, e.g. through the so-called time-delay neural networks [174], or different forms of recurrent neural networks [54, 75, 146]. In recent years, more attention has been directed towards end-to-end learning approaches, which combine acoustic and language modelling in a

single component [7, 17, 40, 41, 44, 74]. Besides the recognition of spoken speech, there is a smaller research community focusing on the recognition of silent speech by analysing the movement of the lips and face, e.g. via electromyography [175, 176].

As computational resources become increasingly available, on the one hand, the recognition performance is further improved, on the other speech recognition is increasingly being introduced to the mass consumer market. Nowadays, most common computer and smartphone operating systems incorporate a speech recognition component, e.g. Apple Siri [208], Google Assistant [223], or Microsoft Cortana [232]. Besides, several companies sell speech recognition components to control connected devices in one's home, e.g. the lights. In 2015, Amazon launched its Alexa system [206], soon followed by Google's Home control devices [224] in 2016. A great challenge besides the recognition of the speech itself is the processing of the transcribed text, i.e. making sense of what the human said and executing commands or retrieving information and giving appropriate replies. A significant difficulty is that the meaning might depend on the context the speaker is in.

1.1.3 Gestures

Gestures are movements of the human body, which serve as a means of non-verbal communication. While the term often relates to hand gestures, it includes all types of movements of the body. Commonly, gestures are not used only by themselves, but in conjunction with speech [173], where they may add additional information, e.g. through pointing at something. The meaning of commonly known gestures strongly depends on the cultural context [9]. The goal of using gestures for human-computer interaction is to recognize gestures performed by the user, which are then mapped to specific commands. The different types of gesture recognition mainly depend on the physical signal they use and where the corresponding sensors are placed, in the environment surrounding the user or on the user directly. Sensors in the environment are less obtrusive. However, the gesture recognition ability may then only be available at the location of the sensor set-up.

One natural option to recognize hand gestures is to place sensors on the hand and fingers directly to track the movements, which is often done in the form of a glove [100, 160, 163]. This approach allows for very exact measurements, may however be considered too obtrusive to the user.

Another way to monitor the movement of the hand (or potentially also parts of the body), is to observe the hand visually using a camera, e.g. employing a smartphone camera [155, 156] or a head-mounted camera. The camera can also be placed in the environment. This setting is also used in several commercial systems, e.g. the Leap Motion [230] for close-range hand tracking, or the Microsoft Kinect [233]. An interesting

technique is presented by Clarke et al. [46, 47], which correlates the motion of a displayed target to the motion the user performs using a webcam. This way, the user is not bound to a specific set of gestures but can follow the targets with any body part. A disadvantage of visual tracking is that the hand (or the respective body part) has to be in the field of view of the camera to be able to recognize the gestures. If the hand has to be held in mid-air for an extended amount of time, this form of interaction might become very tedious. The approach depicted in Figure 1.5 presented by Kim et al. [96], mounting a camera directly on the wrist viewing the hand tackles this problem. However, this is relatively obtrusive.

For dynamic gestures, i.e. gestures which are performed by moving the respective body part, motion sensors placed on that specific part can be used to infer the gesture. The benefit is that motion sensors are relatively inexpensive and also small. Another advantage is that many devices already contain motion sensors, e.g. smartphones or smartwatches. Particularly on hand gesture recognition running on smartwatches, there are several previous works [15, 161, 185, 197], as smartwatches are particularly useful as they are placed close to the source of the relevant motion. A disadvantage is that the motion sensors are not able to measure the finer-grained movement of the single fingers unless sensors are placed there, which results in an approach similar to a data glove, as mentioned above.

Besides measuring the effects of a gesture, either the visual changes or the motion, one can also measure the source of the muscle contraction leading to the movement employing electromyography (EMG). EMG measures the electrical activation of the muscle, which results from a depolarization in the muscle cells caused by the incoming nerve signal. This depolarization, in turn, causes the contraction of the muscle. While there are also medical applications, e.g. detecting neuromuscular abnormalities, several researchers have investigated the use of EMG for interaction [6, 42, 58, 64, 95, 127, 147–149, 151, 198]. Traditionally, EMG measurements require complex hardware. Only recently, different

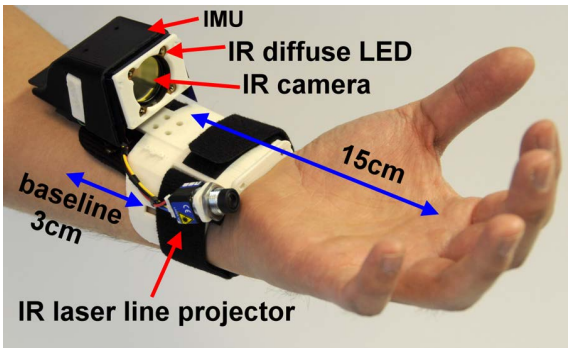


Figure 1.5: The *Digits* system incorporates a camera mounted on the user's wrist and hence can recognize gestures in any pose of arm and hand. Taken from [113].

companies have developed wearable and wireless components, which can be easily used by end consumers [219, 243]. A great challenge when using EMG is that the signal is influenced by noise caused by friction on the skin and its characteristics strongly depend on the specific person.

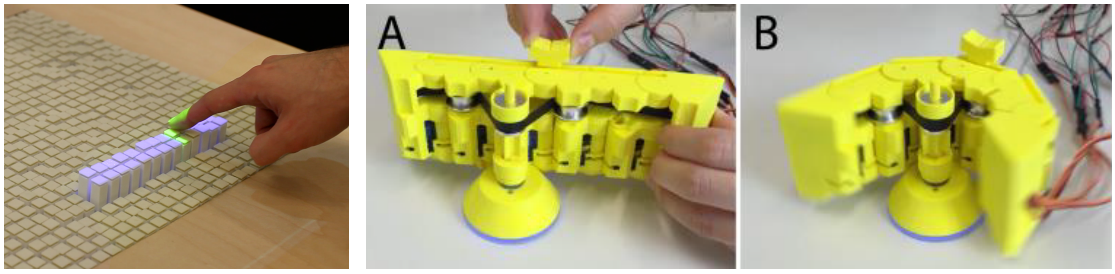
Furthermore, there is a wide range of other methods which can be used to recognize gestures, e.g. the distortion of an electromagnetic field [113, 140] caused by the gesture motion which can be measured with antennas as shown in Figure 1.6, the reflection of radar waves [177] or using sound. When employing sound as the signal the gestures are recognized from, one can either actively emit signals and measure their change to infer the gesture [77, 196], or passively listen to the sounds a gesture itself may generate [78, 108, 109, 194, 195]. The latter concept is included in one of our gesture recognition methods.

1.1.4 Tangibles

Tangible user interfaces (TUIs) are interfaces the user interacts with by touching them and are usually manipulated employing the hand. As mentioned in Subsection 1.1.1, initially, all user interfaces were tangible, as they relied on mechanical controls, such as buttons, knobs, and levers. With the introduction of computer displays, more and more interface functionality was transferred to graphical user interfaces, which have the advantage that the same display can show arbitrarily many different interfaces, which can also easily be updated. However, one loses the haptic feeling of touching an element, which suits the highly advanced fine motor skills of the human hand and provides direct feedback. This property is often valued by users [205]. For this reason, there has been a movement in research back towards tangible interfaces which started several decades ago, and there is continuous and increasing interest in this area. A common theme is to connect virtual

Figure 1.6: The *WiFinger* system recognizes gestures by measuring the distortion in the electromagnetic field between an emitting and a receiving Wi-Fi antenna. Taken from [113].

objects to physical counterparts, described as *Tangible Bits* by Ishii et al. in 1997 [87]. Fitzmaurice et al. [67] presented an example of a tangible interface which incorporated physical bricks. Moving them would lead to the corresponding manipulation of virtual counterparts. A common problem with tangible interfaces is that they usually exhibit a static shape and their representation capabilities are therefore limited. Solutions to this problem entail relatively complex hardware, which can change shape. Examples are an adjustable array of pins, presented by Follmer et al. [68] (cf. Figure 1.7a), which can represent different outputs, but also be used for input by manipulating the pins, or a shape-changing interface which can turn from a slider to a knob [98] (cf. Figure 1.7b). An interesting idea to avoid the need for special-purpose hardware is to employ everyday objects in the environment of the user for interaction. The objects are tracked and can be used as input devices by manipulating them. For example, iCon [43] allows the user to attach markers to physical objects, which are henceforth tracked by a camera, as shown in Figure 1.8.



(a) Taken from [68]. (b) Taken from [98].

Figure 1.7: Two examples of shape-changing tangible user interfaces.

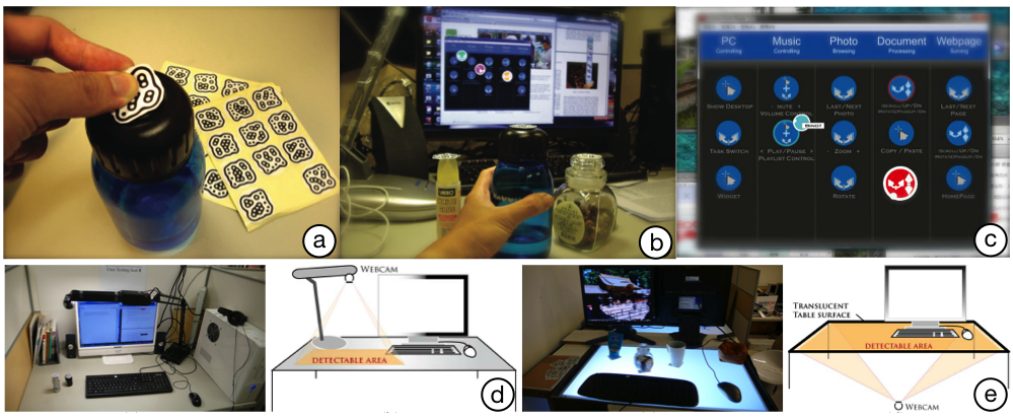


Figure 1.8: The iCon system allows a user to employ everyday objects as tangible user interfaces by attaching markers to the objects through which they are then tracked. Taken from [218].

1.1.5 Gaze

Our eyes cannot only perceive our environment but also be used for outputting information. In human-to-human communication, the eyes transmit important implicit cues, e.g. through eye contact, but also explicit cues, e.g. one can "point" to a particular direction by looking there, which is then observed by others [150]. For human-computer interaction, a natural use case of the user's gaze is object selection, as a human would also naturally look at an object he/she intends to interact with. Eye tracking, i.e. inferring where the user is looking, is a rapidly expanding research area. It can also be employed for psychological analysis, e.g. for marketing purposes by examining how much time users spend looking at different parts of an advert. Eye tracking can be achieved through different means, one of the most common being mounting cameras on glasses, which observe the eye, as shown in Figure 1.9a. Traditionally these systems incorporate infrared emitters directed at the eye and track the reflections of the infrared beam. However, there are recent approaches to estimate the gaze direction from standard camera images only [200, 201]. Another possibility is to employ electro-oculography (EOG), which exploits the fact that the eye is a dipole and changes the electric field around it when it moves, an effect which can be measured with electrodes (cf. Figure 1.9b). Besides, there are stationary eye trackers, which use cameras observing the user as depicted in Figure 1.9c.

In HCI, eye trackers can be used for selection tasks as mentioned above, a concept which is also suitable for assistive applications, e.g. selecting characters on a screen. Such selection schemes fundamentally suffer from a problem called the "Midas" touch, a term which epitomizes the fact what everything one looks at could be selected. This is often solved by a secondary confirmation mechanism such as a button or by waiting for a certain

(a) A wearable eye tracker using infrared cameras to observe the eyes. Taken from [236].

(b) A wearable pair of EOG glasses. Taken from [228].

(c) A stationary eye tracker mounted on a computer screen. Taken from [216].

Figure 1.9: Different eye tracking systems.

"dwell time" before an object is selected. Besides pure direction-based approaches, it is also possible to recognize patterns in the deliberate movement of the eyes, so-called eye gestures, such as moving one's eyes in a circle [13, 59, 128].

Eye motion can be divided into fixations and saccades, which are fast movement between fixations. Another type of eye motion occurs when the user is following a moving object. In this case, the eye smoothly follows the trajectory of the object. Hence this movement is called smooth pursuits. These phases of smooth pursuit can be used for selection. By displaying moving targets on a screen and measuring the eye movement, one can correlate the eye movement to the object trajectories and find the target object as the one with the highest correlation to the eye movement [65, 169].

1.1.6 Brain-Computer Interfaces (BCI)

Instead of observing human inputs from the outside, an intriguing idea is to measure the intention underlying an interaction, i.e. the processes in the human brain. The phenomenon of ongoing electrical processes in the brain was already observed in the late 19th century. The primary measurement categories are divided into electroencephalography (EEG) and the sensing of the hemodynamic activity in the brain. For the latter, there are two methods, functional magnetic resonance imaging (fMRI) or near-infrared spectroscopy (NIRS).

EEG measures the electrical activity of the brain, which results from electrophysiological processes in the neurons during information processing. EEG is usually performed using surface electrodes attached to the user's head. However, the challenge is that the electrodes then only measure the sum of many electric potentials, i.e. the spatial resolution is relatively poor. Moreover, an electrical event will be measured by several electrodes, i.e. it is difficult to find the source. When using surface electrodes, EEG can be affected by electrical fields caused by friction of the electrodes against the head, which creates potentials possibly even higher than the brain signals being measured. Another option is to use invasive electrodes implanted into the brain on the surface of the cortex, which achieves a higher spatial resolution. However, this is currently only suitable for medical purposes.

FMRI and NIRS methods rely on different physical properties of oxygenated and deoxygenated blood. In the case of fMRI, the magnetic properties change, for NIRS the optical properties are different. Changing cognitive loads in a certain brain region cause a change in the ratio of oxygenated and deoxygenated blood in that region. This ratio change makes it possible to infer cognitive activity. Since the response to higher cognitive load in the form of blood flow is relatively slow (in the order of seconds), both methods have a low temporal resolution.

Many use cases of these measurement methods lie in the medical domain, e.g. in the analysis of mental disorders such as epilepsy. Research on actual brain-computer interfaces began in the 1970s. BCIs are often intended to restore capabilities of people impaired by neuromuscular diseases, e.g. paralysis or amyotrophic lateral sclerosis (ALS). A well-known example of a BCI is the P300 matrix speller, presented in 1988 [66]. The letters of the alphabet are shown in the form of a matrix. Specific rows and columns are highlighted in quick succession. The user focuses on the letter he/she wants to write. The method exploits the fact that the highlighting of a row or column induces an event-related potential (ERP), about 300 ms after the event, which can be measured with a surface EEG device. By flashing rows and columns multiple times, the system can infer which letter the user was focusing on. Naturally, a P300 system can also be used for other control tasks than spelling. Other systems allow the user to control a cursor on a screen [126] or even to decode spoken phrases from brain measurements [80]. A challenge with current brain-computer interfaces is the low throughput rates which can be achieved. Nevertheless, they might be useful for implicit interaction, e.g. by being able to measure the cognitive workload as mentioned in the next section.

1.1.7 Passive Interfaces

Most of the interfaces mentioned up to now are active interfaces, through which the human deliberately issues commands or provides input, i.e. interacts explicitly. Passive interfaces try to measure and estimate the state of the human or his/her environment, either for mere monitoring purposes or to adapt other interfaces, i.e. they enable an implicit interaction. For this purpose, different modalities can be used, some of which are discussed above. For example, activity recognition can help inform the user about his/her physical activity, or a smartphone which realizes its owner has returned home might automatically send a message to the home management system to turn the lights on. Human activity recognition has undergone thorough research in the past years [172]. Other use cases are health and fitness tracking through wearable sensors, which can measure the heart rate, electrodermal activity, and oxygen saturation [4]. Many such trackers are already available on the consumer market. Potentially even more interesting for the future are passive interfaces which try to estimate the mental state of the human [105]. For example, the workload (the ratio of cognitive resources employed for a specific task and available cognitive resources) of a user could be measured using electroencephalography (EEG) [20]. As a consequence, working environments could be adjusted, such as changing the complexity of the task representation, to accommodate for a dynamically changing workload. Further possibilities include electro-oculography (EOG) to measure eye movements and for example, to estimate the fatigue of a user [162] or the measurement of the facial temperature to assess the workload level as high workload increases the blood flow to the brain [204].

1.2 Contributions, Positioning, and Outline of the Thesis

Each of the following chapters, except the last, concluding chapter, describes a main contribution of the author. We begin with gesture recognition as a more common interaction modality and continue with more progressive and forward-looking methods of interaction, such as interacting in augmented reality. Each chapter is a self-contained part of the thesis, which may be read independently of the others. The following paragraphs briefly summarize each of the contributions, which are presented in more detail in the single chapters later.

Wearable Gesture Recognition based on Electromyography We start by describing *TouchSense* in Chapter 2, a method employing electromyography to identify the finger used in touches and to estimate the force applied. The reason this is so interesting is that the problem does not concern recognizing standard hand gestures but an action which every one of us does very often every day, touching something, and using this as an opportunity for providing input. If these touches could be measured, they could, on the one hand, be used for active interactions, similar to standard hand gestures to control a device. On the other, we could passively collect information on the user, which might lead to insights on his/her mental state etc.

One option to measure the force of finger touches (however not the specific finger being used) would be to augment the environment and all possible interaction surfaces with force sensors. This augmentation would require a high investment in hardware and installation effort, and in the end, would still limit the interaction surface to a confined space.

For a wearable system, an option to identify the finger used while touching something and measure the force would be to attach force sensors to the user's fingers. However, this is a rather obtrusive method. Instead, we can examine the source of the finger movement, the muscle activation of the forearm. This activation triggers the contraction of the muscles and thereby controls which finger is flexed, and also how strongly. Electromyography (EMG), the measurement of the muscle activation, more precisely the measurement of the electrical potentials which are trigged by so-called motor neurons and in turn trigger the muscle contraction, is usually performed with expensive, stationary hardware, either with needle electrodes penetrating the muscle or surface electrodes applied to the skin.

As it is our intention to create a wearable system usable by the typical user, we employ a recently developed EMG armband, the Myo, containing surface electrodes. The Myo can stream the EMG data to a connected device via Bluetooth, for example, a smartphone.

The Myo itself can recognize five standard gestures and has further been used in research for standard gesture recognition. The question we posed was whether relatively simple hardware such as the Myo could be used to identify finger touches and estimate the force applied in real time, in the light of the few previous research works using traditional complex hardware showing that this was a difficult problem.

We created a hardware set-up with force sensors to measure finger presses with the thumb, the forefinger, and the middle finger and combinations of the three and collected data with this force ground truth from 18 participants. Then, we designed an efficient neural network for classification of the EMG data, which runs in real time on a standard smartphone. While a general user-independent model showed poor performance due to the inter-personal differences in EMG data, we could show that it is feasible to train a user-dependent model for finger identification and provide a method for force estimation. We demonstrate our approach through four applications, which show that it can transform any surface into an interactive surface. It can thereby extend the interactive surfaces of conventional devices, such as tablets, but also create virtual controls where there were none before, e.g. on bicycle handlebars as demonstrated in Figure 1.10.

Figure 1.10: A user controlling the zoom setting of a map application with *TouchSense* without having to let go of the handlebars.

Wearable Gesture Gecognition based on Motion and Sound Sensing Instead of focusing on special kinds of sensor hardware, in Chapter 3, we present *GestEar*. *GestEar* runs on standard smartwatches, relies on their motion sensors and microphone and combines motion and sound information for gesture recognition (the name stems from the fact that an ear also contains audio and motion senses). Motion measurement on wearable devices by itself has often been employed in previous research for gesture recognition. Some gestures, such as snapping, knocking onto something or clapping, also

emit a sound signal along with a motion signal. Interestingly, some of these gestures have been or are still used in many cultures to issue commands, i.e. they could also be used for device control at least for quick and straightforward commands as shown in Figure 1.11. A smartwatch is well suited for the measurement of both signals as it is close to the hand, the signal source, and smartwatches usually contain motion sensors as well as a microphone, which is intended for speech recognition but can naturally be used for our purpose. The question is if and how much the audio information helps the recognition process or whether it may even contribute additional capabilities such as recognizing gestures performed by the hand not wearing the smartwatch just from the sound of that gesture. We collected data from 16 participants for snapping, knocking, and clapping, including samples from both hands for snapping and knocking, and repeated gestures for knocking and clapping, such as clapping twice in quick succession, to also evaluate the recognition performance for these types of gestures. Using recordings from non-gesture activities, we are able to train a robust neural network for gesture recognition, which also inherently fuses the audio and motion information. The model proves to be user-independent with high accuracy, robust to noise, and also runs in real time on a standard smartwatch. When compared to a model which only uses motion, we can observe that the particular advantage of the version with sound is a significantly lower rate of false positive predictions.

Interaction Enabled through Wearable Augmented Reality In Chapter 4, we move from distinct methods for providing input for interaction through gestures to a modality which may also change how the user perceives the smart device. Through augmented reality (AR) provided by head mounted augmented reality displays, we can add interfaces directly in the user's first-person view. We examine how we can use these augmented interfaces for interaction. Physical objects are augmented with interaction

Figure 1.11: A user controlling a light by clapping with a smartwatch running *GestEar*.

widgets which the user can directly manipulate. For example, a lamp may be augmented with controls for brightness and colour, which the user can modify directly in his/her egocentric view, as shown in Figure 1.12. A practical benefit of this concept is that all the user interfaces can be outsourced to the wearable augmented reality representation, i.e. the devices themselves can become simpler. Moreover, each user could personalize their AR representation.

Of particular interest within this interaction concept is how the users can provide input to such a system. We created a prototype implementing our concept, which allows a user wearing a Microsoft Hololens to select a device using his/her gaze, which is approximated by the head pose. Devices are then recognized by their individual markers, and a user interface is generated on demand upon selection and recognition. We carried out a study with 25 participants and studied three different ways of providing input: (1) simple in-air finger pinching and sliding, (2) whole arm gestures such as rotating the arm and waving, and (3) using a physical object as interaction proxies and mapping their movements to interaction primitives.

Especially the third option presents an intriguing opportunity. As we can generate visual widgets on any surface or object, we can augment simple objects with such widgets, instructing the user how he/she may use them for interaction. For example, we can augment a tea mug with the controls for a sound system, informing the user that a rotation of the mug will alter the volume, and linear movements change the track. By tracking the movement of the object, we can then translate the user's actions to device primitives. The fascinating implication is that through the use of AR, we can incorporate simple, non-smart objects into the world of smart devices. Besides, the physical objects allow us to make the interaction tangible.

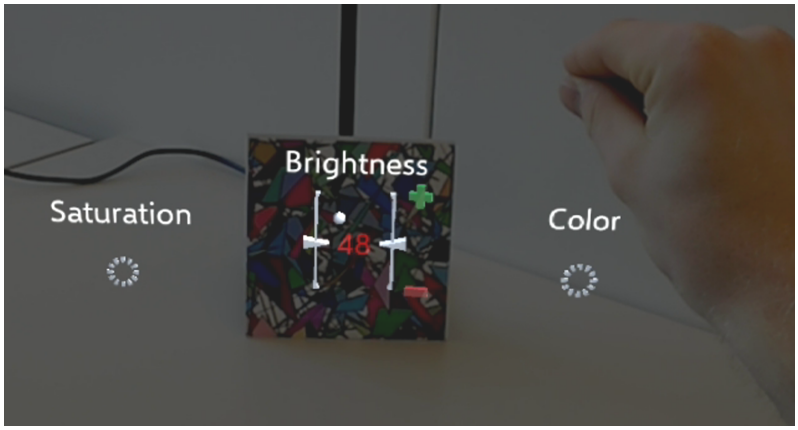


Figure 1.12: A device's controls are directly shown in the egocentric perspective of the user through the use of augmented reality.

Despite this exciting property, we find that the first option is the one preferred by participants, which is founded on a specific characteristic made possible through our concept of interacting in AR: the direct interaction with the devices. The participants felt that in comparison to a standard remote control, for example, they could control the devices directly, because they were constantly looking at the widgets placed on or around the devices instead of a proxy controller such as a remote control or the physical object in the third scenario. This finding reinforces the general idea we propagated at the beginning and which motivates our work, that we should enable an interaction that moves away from proxy controls and enables a more direct interaction. Our AR prototype enables this property. In comparison to our works above which deal only with providing input, it spans several interaction stages: the selection and recognition of devices through the user's gaze and individual device markers and providing input through one of the three mentioned options. Finally, it also avoids having to perform a command inference (i.e. mapping the user's action to a device command) as the user is presented with all interaction choices visually and makes the decision him-/herself. Thus, our work provides a valuable application of AR for interaction.

Besides using AR for interaction with the devices directly, we can also use it to enable easy reconfiguration of smart environments. A typical set-up in the future could be as follows: there are multiple control elements and multiple devices available in the environment which are able to communicate via a common API. The connections between the controllers and devices are not hard-wired (such as it is still common in buildings) but can be reconfigured to the personal preference of the user. We extend our work on AR interactions and enable the user to configure these connections by selecting controllers and devices which should be connected. At the same time, connections are visualized to allow the user to easily understand the control flow (cf. Figure 1.13).

Personalizable Tangible User Interfaces from Simple Material The most significant benefit that the third interaction method using physical objects for interaction in the AR project offered and what participants specifically mentioned is the haptics of the object, which allowed them to perform more fine-grained adjustments. Providing interfaces which are tangible can greatly enrich the interaction experience, as humans are innately used to tangible interactions and also obtain direct feedback through the tangible interface, which enables natural and precise handling. Many research efforts are moving in the direction of tangible interactions, for example, by connecting physical objects to graphically represented virtual counterparts. However, tangible user interfaces rarely adapt to the specific use case or user as the hardware is fixed to a single configuration. There are recently developed shape-changing tangible interfaces, yet these require complex and costly hardware constructions.

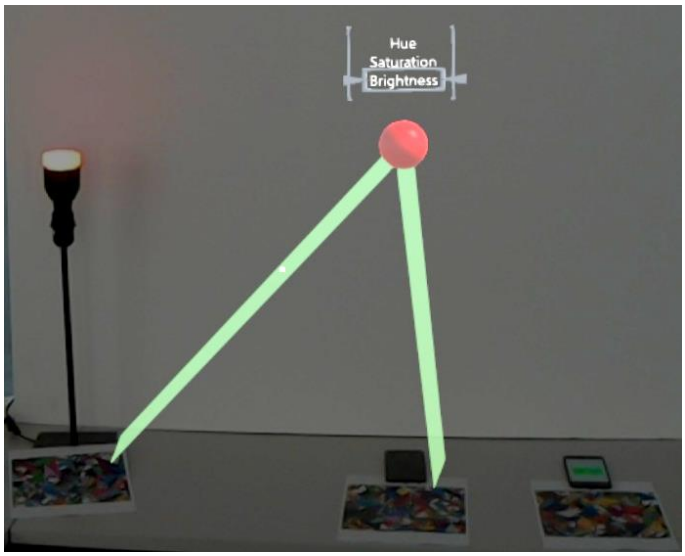


Figure 1.13: We create a system which allows the user to configure device connections in AR.

In Chapter 5, we hence present our approach *Tailored Controls*, which allows users to build their personalized tangible interfaces, created for their current needs from simple materials, such as paper. User can cut out shapes from paper using a pair of scissors, potentially assemble them to more complex groups. Then, they can virtually add specific interaction properties to the shapes, such that a paper snippet is clickable or rotatable (i.e. that rotation is tracked). We carried out a first user study to find the relevant properties, which we call *touchets* (e.g. rotation or that a shape should behave like a button). We developed computer vision algorithms to track the shapes and the user's hand with an RGBD (colour and depth) camera mounted above the interaction surface, as illustrated in Figure 1.14. Thereby, we can observe when a user touches a shape and subsequently enact the interactive properties assigned to that shape. For example, if it is a button and rotation-sensitive, and the user touches a shape, we forward the click event and by how much the shape was rotated to the connected application.

The proposition of our approach is that it enables the user to create universal interfaces for any application, by combining any number of touchets, i.e. interaction patterns, with arbitrary shapes, and connecting them to the application via HTTP requests. Moreover, the user does not require any markers attached to the shapes, which is the case in previous works, and can reconfigure the interface at any time.

In a second study, participants successfully built interfaces for several real applications. A great benefit of our approach is that it enables users to directly see their interfaces in action, which spurs creativity and facilitates fast iterations on interface designs, which was the case for many of our participants. We see interface prototyping, which relies on

this behaviour and usually uses paper but without augmenting it with functionality, as one compelling application area of our approach. Furthermore, a similar system could be used to create more dynamic environments, e.g. for building control. Users could easily create dynamic control interfaces for lights, blinds, the heating etc. without requiring changes in the wiring of the building. Another application area could be factory workers who assemble their personalized interface for machines they work with.

The system we present here is different from our other work mentioned above as it is not wearable and requires a hardware sensor in the user's environment. However, it still follows the general theme of bringing together human users and smart environments by providing accessible interactions, which are comparably easy to use. Moreover, we envision that the concept could also be implemented in a wearable system, as head-mounted displays for augmented reality contain depth cameras to perceive the environment in 3D, which could be used to implement our approach only using wearable cameras.

We conclude this dissertation in Chapter 6, summarizing our contributions, giving an outlook on potential future directions in this and related fields of research.

We want to point out that we acknowledge that our work does not solve the problem of providing humans with natural and intuitive interactions in its entirety. From a schematic point of view, the interaction with a smart device can be understood as a process with several stages, from selecting an interaction target to providing raw input, and understanding the human's intention and inferring an action from the context and that input. This division implies that a solution does not result from a single separate and constrained research field

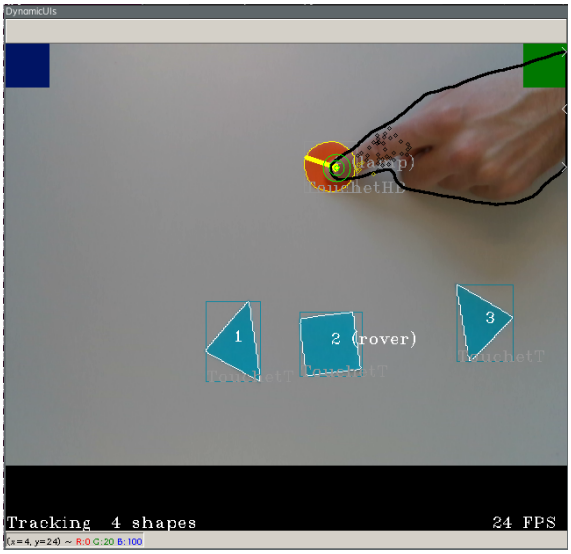


Figure 1.14: *Tailored Controls* recognizes the shapes the users produced and hand interactions upon them. These interactions can be forwarded to any application.

but from a potentially complex combination of results of different contributions. Rather than aiming to resolve the problem in its entirety (which will require many more research efforts), our goal was to provide solutions for sub-problems, e.g. recognizing gestures for providing input. Our solutions could be integrated as a component in a larger system, or to explore paradigms which enable novel ways of interaction such as augmented reality.

Nevertheless, we believe that our research, on the one hand, advances the state-of-the-art by providing novel solutions in areas such as touch recognition and employing augmented reality for direct interaction. On the other, we hope that it inspires future work in particular in the domain of wearable computing and applications thereof by proving that we can run our solutions in real time and use them in real-world applications with the demonstrators we created.

1.3 Contribution Statement

The contributions presented in this dissertation partly rely on work done by other people, who are also co-authors of the corresponding papers. The areas which others contributed to are listed in the following, which mainly consist of implementation work in student projects. The algorithms and concepts implemented are the work of the author. Otherwise, the work presented is also the original work of the author.

Chapter 2: Pietro Oldrati created the hardware measurement set-up for the finger force data collection and collected the EMG data from participants as part of his Bachelor's thesis.

Chapter 3: Linus Fessler implemented the first data collection application for the *GestEar* project, partly collected the gesture data, and built the first gesture recognition prototype in his Bachelor's thesis. The applications were later reimplemented in an optimized fashion by the author, including the processing and gesture recognition (which also entailed significant changes of the model).

Chapter 4: Felix Rauchenstein implemented the prototype of the AR interaction system and collected the participant data. He also implemented the prototype for configuring device connections, which the author reimplemented later.

Chapter 5: Sandro Kalbermatter implemented the prototype for *Tailored Controls* as part of his Master thesis. He also collected the participant data during his project.

Furthermore, the co-authors of the publications mentioned below gave feedback on the concepts, ideas, and evaluations presented in the papers.

1.4 Publications

The research presented in this dissertation has also been published in the following research papers.

2019 [24] **Vincent Becker**, Sandro Kalbermatter, Simon Mayer, Gábor Sörös. Tailored Controls: Creating Personalized Tangible User Interfaces from Paper. *ISS '19*.

2019 [23] **Vincent Becker**, Linus Fessler, Gábor Sörös. GestEar: Combining Audio and Motion Sensing for Gesture Recognition on Smartwatches. *ISWC '19*.

2019 [29] **Vincent Becker**, Felix Rauchenstein, Gábor Sörös. Connecting and Controlling Appliances through Wearable Augmented Reality. *Augmented Human Research, 2019, Springer*.

2019 [30] **Vincent Becker**, Felix Rauchenstein, Gábor Sörös. Investigating Universal Appliance Control through Wearable Augmented Reality. *AH '19*.

2018 [28] **Vincent Becker**, Pietro Oldrati, Liliana Barrios, Gábor Sörös. TouchSense: Classifying Finger Touches and Measuring their Force with an Electromyography Armband. *ISWC 2018*.

2018 [21] **Vincent Becker**. Augmented Humans Interacting with an Augmented World. *MobileHCI '18 Doctoral Consortium*.

2018 [27] **Vincent Becker**, Pietro Oldrati, Liliana Barrios, Gábor Sörös. TouchSense: Classifying and Measuring the Force of Finger Touches with an Electromyography Armband [poster]. *AH '18*.

Besides the papers named above, I authored or co-authored papers during my doctoral studies, which are not part of this dissertations. These are listed below.

2020 [12] Mihai Bâce, **Vincent Becker**, Chenyang Wang, Andreas Bulling. Combining Gaze Estimation and Optical Flow for Pursuits Interaction. *ETRA '20*. The first three authors contributed equally.

2018 [26] **Vincent Becker**, Wilhelm Kleiminger, Vlad C. Coroamă, Friedemann Mattern. Automatically Estimating the Savings Potential of Occupancy-based Heating Strategies. *Energy Informatics 2018*.

2017 [25] **Vincent Becker**, Wilhelm Kleiminger. Exploring zero-training algorithms for occupancy detection based on smart meter measurements. *Computer Science - Research and Development, 2017, Springer*.

2017 [22] **Vincent Becker**, Mihai Bâce, Gábor Sörös. Wearable machine learning for recognizing and controlling smart devices. *MobileHCI '17 Workshop*.

2017 [14] Mihai Bâce, Philippe Schlattner, **Vincent Becker**, Gábor Sörös. Facilitating Object Detection and Recognition through Eye Gaze. *MobileHCI '17 Workshop*.

Two of the earlier publications [25, 26] are related to a project carried out in the area of "smart energy" at the beginning of the doctorate, which also serves as an example for a possible service enabled by ubiquitous computing. A general assumption is that the incorporation of information and communication technology in building control can save a lot of energy. We focused on heating control, as heating has the highest energy consumption for residential households. Furthermore, it is rather challenging to control predictively, due to the significant time difference between control setting and consequence, e.g. setting the thermostat to a higher temperature until the outcome, a higher temperature, is reached, especially when compared to other building control components, such as lighting. The basic idea of how to save energy is to turn off the heating whenever the inhabitants are absent by employing an automatic occupancy control system. However, for some groups of people, it might not be worth the cost of having such a system installed due to no or only short absences. In this case, the household would have to be heated even during the absences to preheat the dwelling for the return of the inhabitants. Besides, very well insulated houses also do not benefit from such a control system as much as houses with little insulation. The goal of our project was to provide a method to automatically estimate the utility of a heating control system for a specific household without installing any additional hardware except a smart meter, which many households already have installed anyway. We detect the occupancy pattern from the smart meter data [25] and based on this schedule we use a heating simulation parametrized to the specific household's characteristics to estimate the heating energy consumption [26].

Another early workshop paper [22] describes a prototype to detect smart devices visually using a smartphone camera and subsequently being able to use gestures to control the device. This prototype is the beginning in the line of work described in this thesis and

an example for the first necessary step in the interaction process, identifying the device the user intends to interact with. This work is not part of the thesis as the thesis focuses on how to provide input to a smart device and not how to identify it. Furthermore, the approach employing visual object recognition has several shortcomings, such as not being able to distinguish different objects of the same kind, e.g. several lights. A similar problem occurs when two users want to interact with two devices of the same type at the same time. There are other solutions available avoiding these problems, such as marker- or barcode-based recognition, or using infrared emitters on the humans and receivers on the devices to exchange identifying information. Nevertheless, these have the disadvantage of requiring the instrumentation of the environment.

Our latest paper [12] deals with a gaze interaction method called *pursuits* (which is also shortly mentioned in Subsection 1.1.5). When the human eye follows a moving object, it carries out a smooth movement, called smooth pursuit, in contrast to the usual saccadic movements. These pursuits can be employed for interaction by displaying moving targets on a screen, measuring the eye motion with a tracking device, and finally correlating this motion to the motion of the targets to select the target the human was following as the target with the highest correlation. One advantage in comparison to other gaze-based interaction methods is that it does not require any calibration (e.g. of an eye tracker), as long as the relative movements of the eye are captured. The interaction technique was presented in 2013 by Vidal et al. [169] and has since then spurred further research both in improving the method itself but also in applications. However, all previously presented works required either a dedicated eye tracker or a camera placed very close to the eye. We propose a method that only requires a standard webcam, which is placed at a distance from the user, for example, mounted on a computer screen. On the camera frames, we find and crop the face and run two algorithms independently from one another. The first is a state-of-the-art gaze estimation algorithm to infer the user's gaze direction, which we translate to the point of regard on the computer screen. These points are then correlated to the positions of the targets. The target with the highest correlation is selected as a first candidate. The second algorithm is an optical flow algorithm, applied to consecutive pairs of image frames, which computes the motion between the frames, in our case, the motion inside the cropped eye patch. This eye motion is compared to the targets' motion (the difference between two target positions in time) employing the cosine similarity. Similarly to the first part, we select the target with the highest similarity as the second candidate. We finally merge the two candidates using a voting strategy. Through this combination, we exploit both the direct gaze information as well as the dynamic information provided by the optical flow. In experiments, our method is competitive with a commercial eye tracker for a small number of targets and certain target trajectories. We believe that our approach can inspire new pursuit interaction approaches which only requires hardware as simple as an RGB camera, which is already available in a large number of devices.

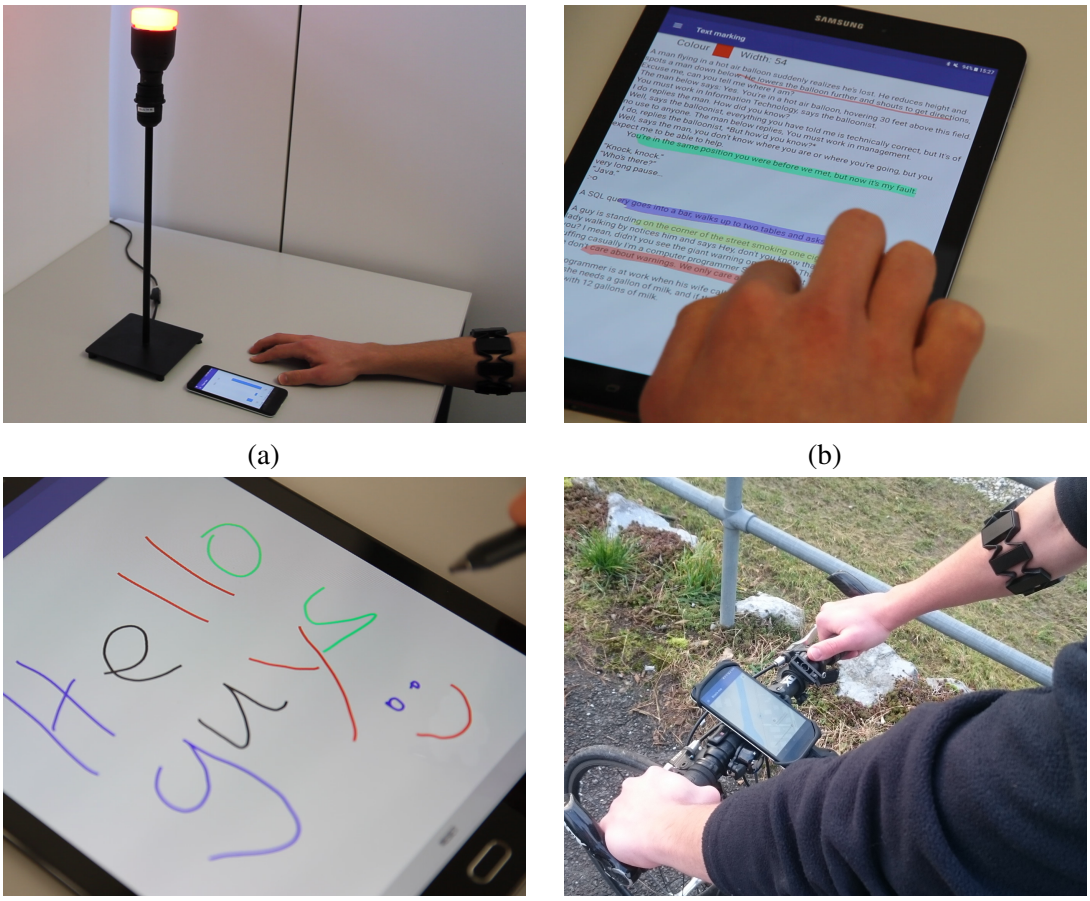
Chapter 2

Classifying Finger Touches and Measuring their Force with an Electromyography Armband

2.1 Introduction

Finger touches, presses, and grasps are among the most important ways humans interact with their surroundings. Evolutionary biologists speculate that the ability to manipulate objects is to a large extent the reason for the remarkable development of primates' brains, which ultimately made humans the dominant species of the planet [63]. Tools built throughout human history take advantage of the broad spectrum of human touch. A piano, for example, can only be meaningfully played with several fingers at once, and its keys react to the force applied with differently loud tones. This richness of human touch, however, is only poorly represented in most technological devices. A traditional light switch, for example, is insensitive to both the pressure of the touch, and to the finger(s) used in the process. It will uniformly turn on the light, with the same colour and brightness. Taking the pressure of the touch gesture into account already yields more sophisticated devices, such as dimming light switches. Considering also the specific finger used in the interaction can add new and more features, and perhaps open entirely new interaction possibilities as additional information can be encoded through the specific fingers.

The lack of adaptation leaves space to explore the use of techniques that identify the finger and estimate the force (or pressure) of finger touches. Such techniques would enable both explicit and implicit input. Explicitly by actively modulating one's finger force when



(a) (b)

(c) (d)

Figure 2.1: We propose a method that allows determining the finger and the force applied in touches. We present several applications using this method to enrich the interaction with devices.

controlling appliances, for example, by determining the brightness of a light by the force of the finger pressing when turning it on. Implicitly by gathering information about the user's state, such as him/her being in a particular mood which can be inferred from the force he/she applies.

Nevertheless, even amongst the newer touch-based devices such as tablets and smartphones only very few feature a pressure-sensitive input surface which does not have the means of identifying the finger either. Apple's newer products incorporate the technology "Force Touch" [247], which can distinguish between different degrees of force being applied to the screen, thereby adding an input dimension.

Instead of augmenting specific devices with such sensing capabilities, it is beneficial to augment the humans themselves to sense the fingers used and the corresponding force of

their touches. That would allow any surface on any object in the environment to act as an input interface and enrich the human's interaction capabilities. For example, one could turn on the lamp by pressing on a surface nearby, such as a wall or a table, or control the TV from the sofa by touching the armrest. For such a system augmenting the human to work, there are two options. One would either have to augment the user's fingertips with force sensors, which is rather obtrusive or measure the contraction of the muscles in the forearm to infer the finger used and estimate the force exerted by that finger, since the muscles in the forearm control the fingers. The latter is possible using electromyography (EMG), the measurement of muscle activation potentials. EMG devices are usually costly and require large amplifiers; however, in recent years, an inexpensive and wireless EMG armband, the Thalmic Labs Myo [243], has become commercially available[1]. Figure 2.2 shows the difference between the Myo and a traditional EMG recorder.

(a) (b)

Figure 2.2: The wireless Thalmic Labs Myo (picture taken from [238]) compared to a traditional EMG recording set-up.

We present *TouchSense*, a system for classifying finger touches of the thumb, forefinger, and middle finger and estimating the exerted force. We only utilize the Thalmic Lab's Myo EMG armband and a standard Android smartphone which continuously receives the EMG signal via Bluetooth, making our system wireless and mobile. For finger identification, we use a light-weight neural network running on the smartphone with a short average inference time of under 10 ms, thus fulfilling real time requirements. Furthermore, the network has a model size of less than 140 KB. Subsequently, the force is estimated and output on a continuous scale. We collected the necessary data from 18 participants and measured force ground truth values with a self-built hardware set-up. This ground truth

[1] At the time of writing of the thesis, the production of the Myo has been discontinued. Nevertheless, other, more advanced products are becoming available, such as the CTRL-kit from CTRL-labs [219].

enables us to train a regressor for actual force values. We evaluate our system in several experiment designs, including a user-independent design, a user-dependent but session-independent design, and a session-dependent design. Moreover, we demonstrate its use in four demo applications. Overall results show that a user-independent system is difficult to achieve, however, a user-dependent, but session-independent system is possible, i.e. our system has to be trained once for every user.

2.2 Background and Related Work

Electromyography (EMG) measures the electrical activation of a muscle. Motor neurons control muscle fibres by electrical impulses. In order to command a muscle to contract, the brain sends an electrical impulse to the motor neuron, which in turn transmits an electrical impulse to the muscle it controls. Following the activation, ion channels open and create a measurable difference in the potential of the muscle cells. Thereby, each muscle contraction can be associated with an EMG signal. A strong muscle contraction is accompanied by a strong EMG signal, i.e. stronger presses should be distinguishable from lighter presses. EMG can be performed invasively, applying needle electrodes to penetrate the muscle, an accurate but impractical way of measurement, or using surface electrodes, which are easier to use but more imprecise. We use the Myo armband, a surface EMG device applied to the forearm, thereby measuring the muscles controlling hand and fingers. A significant challenge when performing EMG analysis is the substantial variation between people due to interpersonal differences in anatomical properties, such as muscle strength, the position of bones, and skin conductance. Furthermore, the measurements of two sessions for the same person may differ because of the exact placement of the electrodes changes.

Most previous work using EMG in the HCI domain has been done in the area of hand and finger gesture recognition [1, 5, 6, 31, 32, 38, 42, 58, 64, 72, 73, 94, 95, 114, 127, 130, 137, 144, 147, 151, 168, 198, 199] and it has been shown that this can be performed at high accuracy using machine-learning techniques. However, the mentioned works investigate only full-hand gestures or coarse-grained finger gestures. Moreover, many previous works differ from ours by either using mostly expensive, specialized, wired, or custom-built hardware for measurement, some with a high number of channels, performing an offline analysis, and not being able to run in real time, or combining EMG with other sensors.

Previous Work on Touch and Force from EMG There is relatively little work considering finger touches and the applied force. In the following, we briefly discuss previous research concerning finger touching gestures and the estimation of applied force.

DiDomenico et al. show the feasibility of using EMG for finger strength regression for the purpose of evaluating the ergonomics of finger-intensive tasks [56]. They collect force ground truth in an experiment, including 30 participants performing finger gestures (e.g. pinches) simulating hand-intensive tasks. The EMG data were gathered using three wired electrodes attached to the forearm. For regression, they fit linear models and show that this results in an acceptable error. However, they do not use the models on unseen data in a test case and only perform an offline analysis.

Saponas et al. [148] classify finger gestures engaging the fore- and middle finger, extended or curled, tap, and lift and also classify finger strength into hard and light. They show that it is possible to determine the finger which is used for pressing and distinguish the two pressure classes with high accuracy. However, they trained and tested with data from the same session. When performing a cross-user-validation, the performance decreases significantly. While they only classify into two force levels, we envision a system which estimates a continuous force level. Their EMG electrodes are wired to an expensive measurement set-up with a high sampling rate of 2048 Hz (more than ten times higher than for our device). Moreover, they only perform an offline analysis. They continued their work and also built an online system for pinching gestures [149] using a custom-built wireless EMG armband. They show that it is possible to obtain over 70% accuracy in a two-session experiment design for pinching gestures using the fore-, middle, and ring finger. We go further and perform the analysis directly on a smartphone, thereby creating a completely mobile system using off-the-shelf hardware.

Benko et al. have built on the work done in [148] to perform finger identification and additionally estimate the finger pressure by using a smoothed average of all EMG channels [32]. They use this pressure estimate in combination with the touch-sensing capabilities of a tabletop computer to allow extended input, such as adapting the width of a stroke in a painting application according to pressure. However, as they use the same system as in [148], their set-up is also static and processed on a desktop machine. Furthermore, their design is session-dependent (i.e. it needs to be trained for each participant before every use and not only once per participant). As they never collected force ground truth, it is also not possible to say how well their estimate correlates to the actual force exerted by the finger onto the screen. In contrast, we collect ground truth data using our hardware set-up and fit a regressor to the EMG data and the ground truth.

Touch and Force through other Modalities Various other modalities have been examined for achieving finger identification. Vision-based approaches employ cameras viewing the scene from above tracking the users' fingers [50, 119, 129, 203]. Others view transparent displays from below and recognize individual fingerprints [85]. While these approaches have the benefit of not having to augment the users with any sensors, the

interaction is limited to a fixed space. Furthermore, it is not possible to determine the applied force.

Contrarily, other researchers attached sensors directly to the fingers or integrate them in a glove to identify which finger is used in an interaction [70, 76, 120, 121]. This method allows for highly accurate finger identification. However, the force of touch interactions is not estimated. Most importantly, they are rather obtrusive systems in terms of finger interaction. Another solution is to incorporate RFID tags into fake fingernails, which are detected by RFID readers in the interaction devices [166]. While this provides a method which does not interfere with the finger interaction, it requires the interaction surfaces to contain an RFID reader. Besides, force estimation is not possible.

Concerning the measurement of force, manufacturers of commercial products focus on hardware solutions in touch screens, as the aforementioned *ForceTouch* from Apple, or touchpads, such as Synaptics' *Force pad* [241], a laptop touchpad which is able to measure the force applied by the fingers. However, the input surface is confined to the relatively small size of the touch screen or touchpad, whereas our system can turn any surface or object into an input surface.

In conclusion, the main differences to previous work are the following: we use an inexpensive, wireless, off-the-shelf EMG armband with a low sampling rate for finger classification and force estimation; we collect force-ground truth for finger presses and are thereby able to evaluate our force estimation and finally we run an online, real-time analysis on a smartphone, through which our system becomes mobile.

2.3 The TouchSense System

2.3.1 Data Collection

To record EMG data, we use the Myo armband from which we deliver the EMG data to an Android smartphone via Bluetooth. Additionally, we collect ground truth force data using a measurement set-up shown in Figure 2.3. This ground truth allows us to train regressors for the exerted strength and the force values also serve as labels for classification (then reduced to zeros and ones). The set-up consists of a measurement circuit using three adjustable force-sensitive resistors wired to an Arduino Yún, which is itself connected to a computer. The force-sensitive resistors are fixed to a board by Velcro pads, in order to adjust them to each participant's hand anatomy. A diagram of the set-up and the schematics are depicted in Figure 2.4a and Figure 2.4b. The smartphone and the Arduino are synchronized via an NTP time server. Note that we only require the

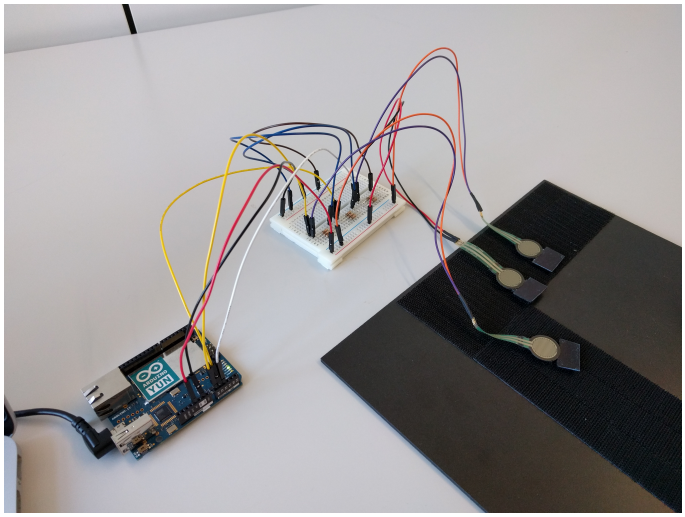
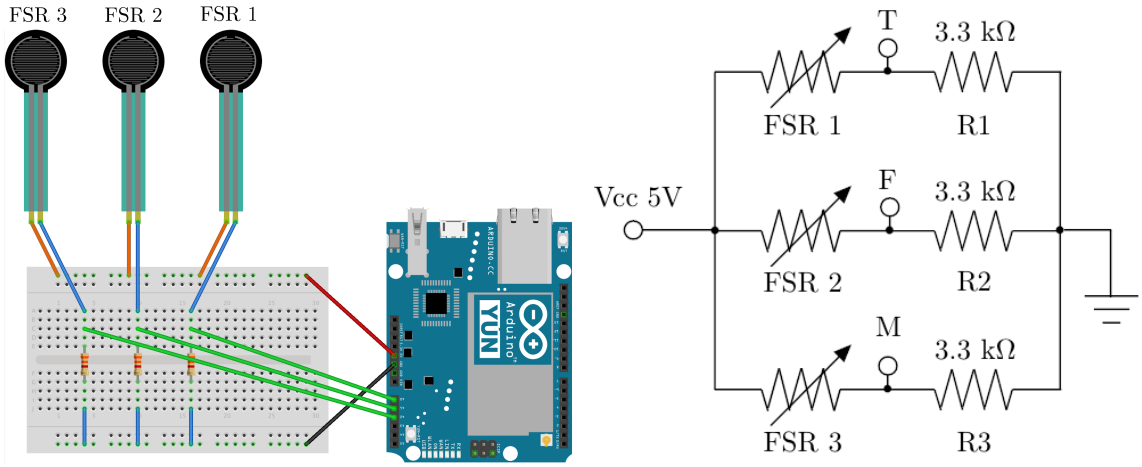


Figure 2.3: Our hardware set-up for collecting force ground truth consisting of a force-sensitive resistor for each of the three fingers (thumb, forefinger, and middle finger) and an Arduino.



(a) (b)

Figure 2.4: A diagram and the schematics of the hardware set-up for measuring the force values of finger presses showing the connections between the force-sensitive resistors and the Arduino.

measurement set-up for collecting training data and not at test time or in a real application. We convert the force signal to Newtons (resolution 0.01 N). Typical pressing forces that can be comfortably exerted range from 0 to 12 N. The Myo samples at a rate of 200 Hz from eight channels (i.e. eight sensors) in a value range from -128 to 127 (unitless). The Myo's sampling rate is relatively low when compared to standard EMG measurement devices (sampling rate commonly around 2 kHz). Potential line noise interference at the frequencies 50 Hz and 60 Hz is automatically filtered. We also set the sampling rate of our force ground truth measurement set-up to 200 Hz.

Each session consists of a series of presses, with soft (around 2 N to 4 N) and strong presses (around 8 N to 12 N), and a period where the participant modulates the pressing force from low to high (i.e. from 0 N to 12 N). This is done for the thumb, the forefinger, the middle finger, and also all combinations of the three. Because we also record combinations, we only included three digits in the data collection in order to keep sessions short. A single session takes around five to seven minutes. We segment every recording in time windows with a length of 10 EMG samples with an overlap of 9, i.e. every window represents a period of 50 ms over eight EMG channels and for every 5 ms of recording we produce a new window. This way, we generate a high number of training samples, even for short data collection trials.

We collected data from 18 participants (six females, 20 to 66 years old, average age 27 years). During data collection, the participants were sitting at a table. We tried to keep the placement between different sessions and participants as similar as possible by placing the Myo notification light on the upper side of the forearm as recommended by the Myo developers. In total, we gathered 1,819,846 samples. All our participants wore the armband on the right arm. However, it should be simple to integrate also the left-handed case by mirroring the EMG channels [95]. For fifteen participants, we collect a second EMG session for a user-dependent, but session-independent experiment design. For two of these, we recorded another seven sessions, i.e. nine in total, to examine the effect of utilizing an increasing amount of training data. For all the participants with at least two sessions, the recordings were carried out at least one day apart from each other. As mentioned above, our dataset also includes touch and force data for any combinations of thumb, forefinger, and middle finger which were collected in the same sessions and the same way as the single touch gestures. As the finger identification task is very challenging even for single fingers, the data for the finger combinations are not used here. A preliminary evaluation including all combinations of the three fingers resulted in a relatively low accuracy of 46.1% for a user-dependent, but session-independent experiment. Nevertheless, the data are included in the dataset we publish for potential future use.

2.3.2 Finger Classification and Force Estimation

As dataset for the classification task, we use all samples with forces over 2 N for a single finger and under 1 N for the other two fingers to remove finger combinations and converted the force values into classification labels. In most previous works based on EMG, researchers have used hand-crafted features for classifiers such as Support Vector Machines. In the first experiments using the data of all participants, we evaluated several approaches with several different sets of features gathered from the literature. However, although we tested a vast set of combinations of features, we could not exceed a test accuracy of around 65% on our dataset. Thus, we decided to design a neural network for finger classification with the intention that the network would learn the best features, relieving us from the requirement of performing feature engineering and selection, an approach recently also followed by other researchers [60, 61].

Our architecture is depicted in Figure 2.5. Since hand-crafted features are mostly computed per EMG channel, we decided to do the same. The first network layer learns features per channel. Since we want it to learn the same features for each channel, the weights of the corresponding neurons are shared. This procedure has the beneficial property of being the same as a convolution over an image with a $1 \times w$ filter without padding, where w is the width of the image. In our case, the "image" consists of eight EMG channels with a window of 10 values each. Hence the filters have the shape 1×10. The result of this first layer is k 8×1 feature maps, where k is the number of filters. Keeping k variable allows controlling the complexity of the network. To reduce the number of feature maps, we add a convolutional layer with l 1×1 filters (l should be smaller than k). These take all k features from the previous layer into account and produce l higher-level features.

The feature maps are then reshaped into a single vector which is the input to the rest of the network. It consists of a fully connected layer with m units, three fully connected LSTM (Long Short Term Memory) cells, also with m units each, and a softmax layer to output the probabilities for each finger. In the fully connected layer, we use dropout with a dropping probability of 0.5. The fully connected layer can take every combination of features resulting from the convolutional layers into account. Being able to learn these combinations automatically is a great advantage compared to other classifiers, where these combinations of channel features have to be encoded in specific multi-channel features (c.f. for example [58]). The motivation for the LSTMs is that they allow us to take information over several windows into account, thus allowing to exploit both temporally local information from single windows as well as longer dependencies. We train the LSTM cells on sequences of five windows and during testing also feed sequences of five windows. The number of outputs of the softmax layer corresponds to the number of fingers taken

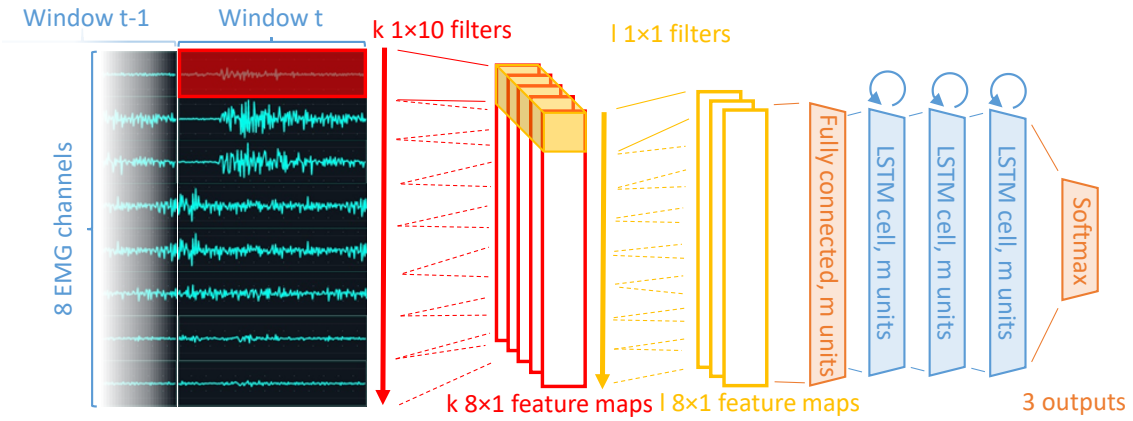


Figure 2.5: Our architecture for finger classification consisting of two convolutional layers with k and l filters, one fully-connected layer with m units, three stacked LSTM cells with m units each, and finally a softmax layer. The LSTM cells take five consecutive windows into account.

into account in the specific experiment, i.e. there are either two or three. The number of feature maps in the convolutional layers and the number of units in the LSTM cells are given as variables to control the complexity of the network to adapt to different amounts of available training data. We use a cross-entropy loss function, an Adam optimizer with a learning rate of 0.0025 and a batch size of 200. The number of training epochs varies for the different experiments, to control for overfitting. We implemented the network in TensorFlow [242], trained it on a computer and exported the model file to our Android application. The model size is only about 140 KB. The runtime for all our models is below 10 ms on average on an LG Nexus 5X and hence fulfils real time requirements for human-computer interaction. We smooth the predictions using a majority vote over the last five predictions to make the results more stable. In the evaluation, this is only done when we test on whole sessions (either in the user-independent or the user-dependent, session-independent setting), as the smoothing requires time-continuous sequences.

To obtain a force value for the finger press, we use the mean absolute value over all channels and the whole window (MAV) as calculated by $MAV = \frac{1}{8*w}\sum_1^8\sum_1^w |v_{ij}|$ where i indexes one out of the eight channels and j indexes one out of the w samples per window. This is a good indicator of force (cf. Section 2.4), which does not have to be learnt. As the predictions, we also smooth the MAV to make the values more stable. If we allow user-dependent calibration, we can also fit a regression function. We use a linear regression model solely based on the MAV to map the MAV to actual force values. The code and data are available on GitHub [214].

2.4 Evaluation

2.4.1 Finger Classification

We carry out several experiments, including mixing all data and performing cross-validation, a user-independent, and a session-independent design. For all our experiments, the description, including the network configuration, the evaluation method, and the average accuracy is given in Table 2.1. Whenever applicable, we perform 30-fold cross-validation. Otherwise, we apply cross-validation across participants or multiple sessions of the same participant. For the user-independent (no. 2 and 5) and the session-independent experiments (no. 3 and 6), we apply prediction smoothing as mentioned in Subsection 2.3.2. For the other experiments, the data is shuffled; hence it loses its sequential character and smoothing is not applicable. We always balance the data for training, i.e. we use the same number of samples per class to avoid skewed data proportions in the training process. Whenever possible, we also give more detailed performance figures in the form of confusion plots and accuracy charts. The confusion plots display the normalized confusion matrix (all numbers divided by the total number of samples), which in case of experiments no. 2, 3, 5, and 6 is the average of the normalized confusion matrix for each participant (no. 2 and 5) or session (no. 3 and 6). The accuracy charts display the accuracy when testing on a single participant in the user-independent experiments (no. 2 and 5) or the average test accuracy per participant for the session-independent experiments (no. 3 and 6).

Expt.	Setting	Cross-validated	Fingers	Configuration	Accuracy
1	All data mixed	30-fold	t, f, m	64, 8, 256	97.4%
2	User-independent	Across users	t, f, m	64, 8, 256	48.4%
3	User-dependent, session-independent	Across sessions per participant	t, f, m	32, 4, 32	72.6%
4	All data mixed	30-fold	t, m	64, 8, 256	98.7%
5	User-independent	Across users	t, m	64, 8, 256	70.1%
6	User-dependent, session-independent	Across sessions per participant	t, m	32, 4, 32	86.8%

Table 2.1: The description of the experiments including the cross-validation, which fingers were included (t: thumb, f: forefinger, m: middle finger), and the network configuration (*k, l, m*). The results are given as the average accuracy over all folds and participants.

Classification with Three Fingers

First of all, we perform our evaluation on all the data (excluding finger combinations as mentioned above), including the thumb, the forefinger, and the middle finger.

Experiment 1 In a first experiment, we evaluated the setting of mixing the samples of all users and performing cross-validation. Due to the large number of samples, we deploy a relatively complex network. We train for 100 epochs and obtain a test accuracy of 97.4%. The confusion plot in Figure 2.9a shows how few misclassifications there are. This proves that our neural network is able to capture the characteristics of the EMG signals.

Experiment 2 In a second experiment, we attempted the most challenging setting, cross-validation on users, i.e. excluding a single participant's data from the training set and testing on this data to evaluate how user-independent our system is. Unfortunately, we obtain poor results for most participants, as shown in Figure 2.6, and confirmed by the confusion matrix (c.f. Figure 2.9c). This outcome is a result of the inter-personal anatomical differences and that the sensors are placed a little differently for every participant. It shows how individual the EMG data of each participant is. The user-independent design is generally a challenge in previous literature as well. The performance is generally poor; however, our primary goal and measure is high performance in the session-independent experiments.

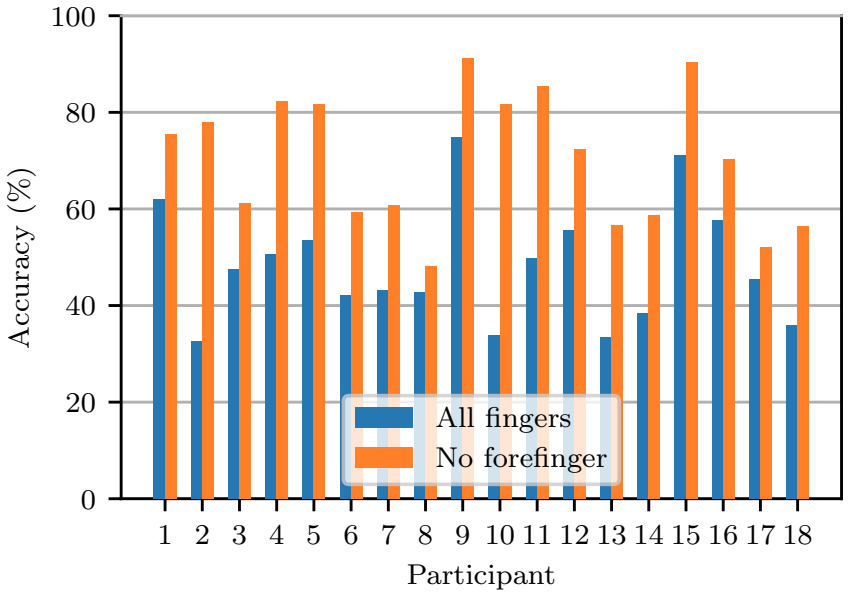


Figure 2.6: Accuracies for participants in experiments 2 and 5.

Experiment 3 As a consequence of experiment 2, we moved to a user-dependent, but session-independent design in the third experiment, which resembles a real-world scenario where the system has to be trained once per user. For this purpose, we recorded at least two sessions for fifteen participants. We perform cross-validation on the sessions, i.e. we test on each session, after having trained on all the others. Note that for this experiment, we have much less training data, so we reduced the training epochs to 20 and also the layer sizes as shown in Table 2.1 to avoid overfitting. The results (c.f. Figure 2.7 and confusion

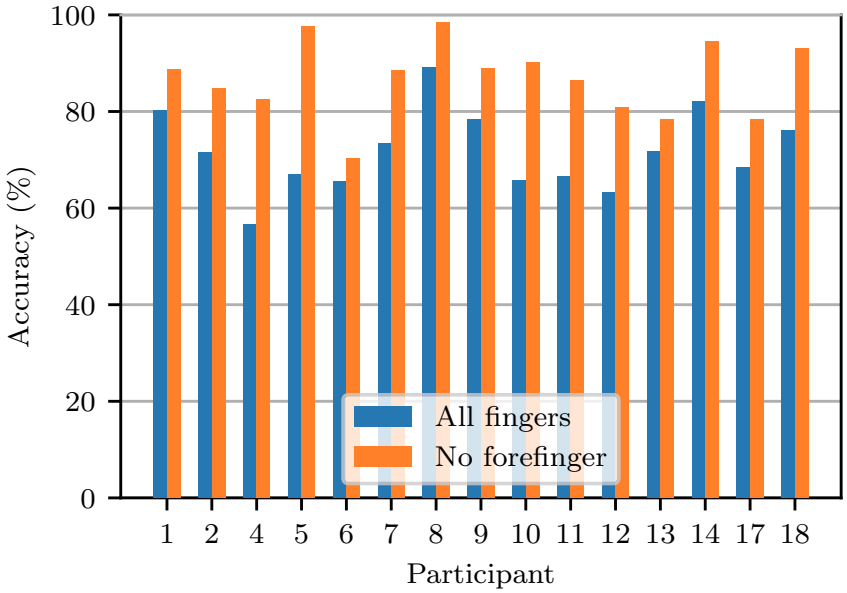


Figure 2.7: Accuracies for participants in experiments 3 and 6.

matrix 2.9e) are much improved from the user-independent setting, as the variance now is only caused by differences between sessions, but not between people. As the sessions were at least a day apart, this proves that we can generalize over a longer time period. As our evaluation for the limited set of two participants with nine sessions each show (cf. Section 2.4.1), we can expect to obtain better results with more data per participant.

Classification with two fingers, experiments 4 to 6

A problem we identified and which is shown clearly by the confusion matrices in Figures 2.9a, 2.9c, and 2.9e is that the forefinger is often confused with either the thumb or the middle finger. We thus also investigated the case of only classifying thumb and middle finger and reran all our previous experiments. We show the result in the confusion matrices in Figures 2.9b, 2.9d, and 2.9f and as second bars in the accuracy figures. As expected, the results for this case are better than in the three-finger case for all experi-

ments. Especially experiment 6, with over 86%, shows that our system can be valuable for real-world applications.

Finger Classification with more Data

As mentioned above, for two participants, we collected nine sessions to investigate the effect of training with more data on the classification accuracy in the session-independent case. In Figure 2.8, we show the accuracy when training on an increasing amount of sessions and performing cross-validation for each of the two participants (including all three fingers, i.e. resembling experiment 3). With more sessions used for training, the performance significantly increases for both participants. We believe that we could achieve higher overall performance with more training data for each participant.

2.4.2 Force Estimation

As mentioned in Subsection 2.3.2, we use the mean absolute value over all channels and the whole window (MAV) as default force estimation. The Pearson's correlation coefficient of the force ground truth value and the MAV for all participants is 0.86 on average, which proves that the MAV is a good proxy for the force and changes in the MAV correspond to changes in the force. The individual correlation values for each participant are given in Table 2.2. For participants with more than one session, we average the results over all of them.

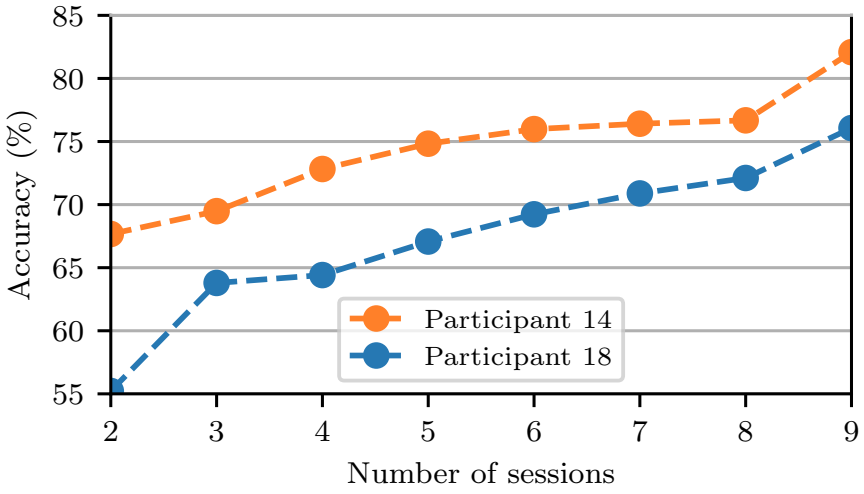


Figure 2.8: The accuracies in the session-independent experiment for two participants when increasing the number of sessions for training.

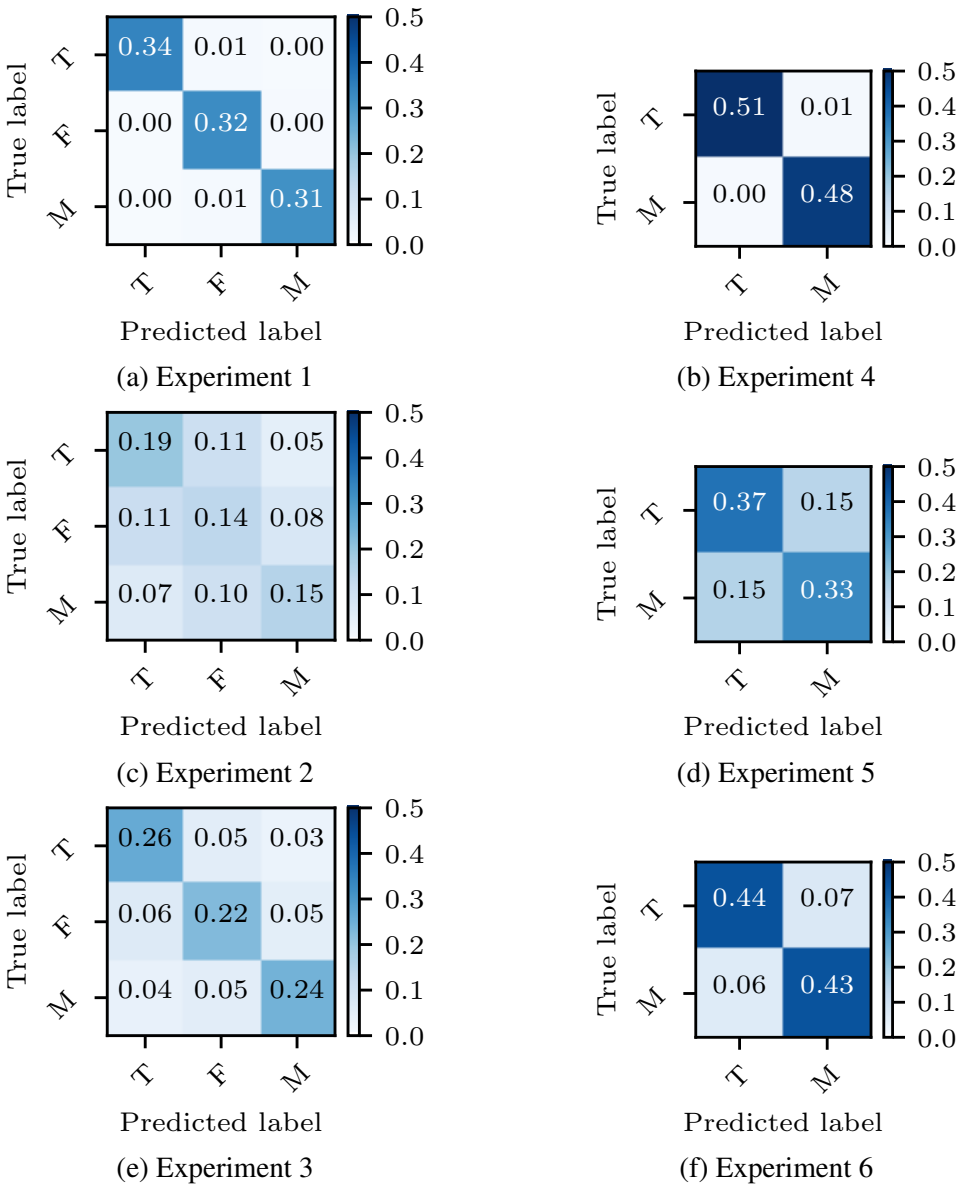


Figure 2.9: Confusion plots for all the experiments.

Table 2.2: The Pearson's correlation coefficient between the MAV and the force ground truth for each participant. The high correlation proves that the MAV is a good proxy for the applied force.

Participant	1	2	3	4	5	6
Pearson c. c.	0.86	0.83	0.83	0.91	0.92	0.84
Participant	7	8	9	10	11	Mean
Pearson c. c.	0.88	0.87	0.77	0.85	0.91	0.86

Nevertheless, if data for calibration is available, e.g. in the multi-session setting, we can fit a regression function. For each participant, we fit a linear regression model based on the MAV and cross-validate it (10-fold). For participants with multiple sessions, we average their results again. The average mean absolute error (MAE) across all participants is 1.26 N, i.e. roughly 10% of the usual force range, which reaches up to 12 N. When smoothing the results over five predictions, as we do it also for the classification, the MAE decreases to 1.22 N. Figure 2.10 depicts the ground truth force values, the MAV (scaled by a factor of 50), and the predicted force values from a linear regression model for a segment of a session with 10,000 samples. It exemplifies that the MAV reflects changes in the force well. Nevertheless, the regression model performs better, especially in areas where no force is applied. Here the MAV is still positive, which results from sporadic EMG activation. For all participants with two sessions, we fit a linear regression function on one session and applied it on the other (and vice versa) without smoothing. The average MAE for this session-independent scenario is 1.28 N, which shows that the calibration of the regression system works across multiple sessions. The results of this evaluation are given in Figure 2.11.

2.5 Limitations

The user-dependence still is a strong limitation, i.e. our system has to be trained per user. Besides, to be able to train the force regression, ground truth force values are necessary, which we measure with our hardware set-up. However, this is not available everywhere, and it does not appear reasonable for every user to have one. Nevertheless, the finger classification could be trained without the exact force information only by instructing the user to perform a series of finger presses and recording the data. This approach would not require any hardware apart from the smartphone and the Myo, which are anyway needed for the normal use. Furthermore, the user could still obtain a force estimate through the MAV, which does not have to be learnt from training data. The user-dependence might

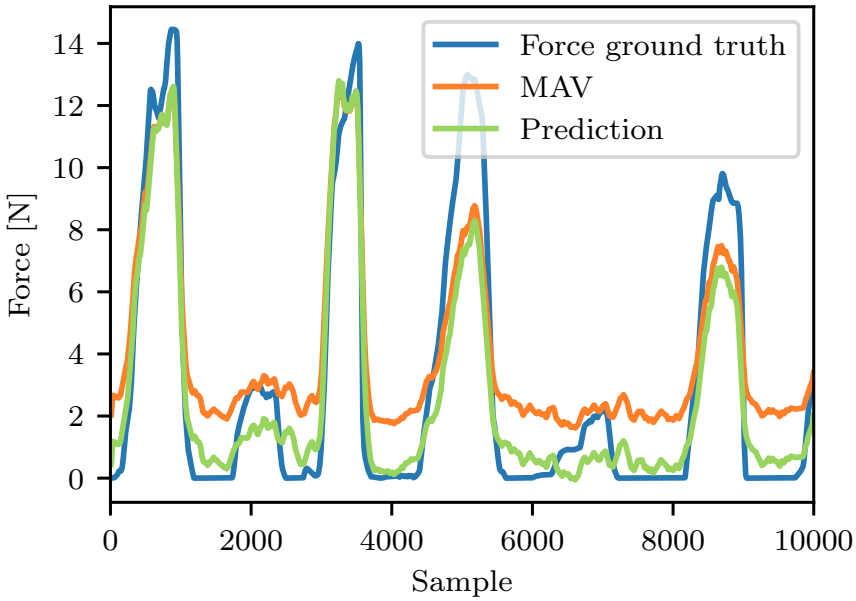


Figure 2.10: An example of the ground truth force values, the MAV, and the predictions of a linear regression model for 10,000 samples.

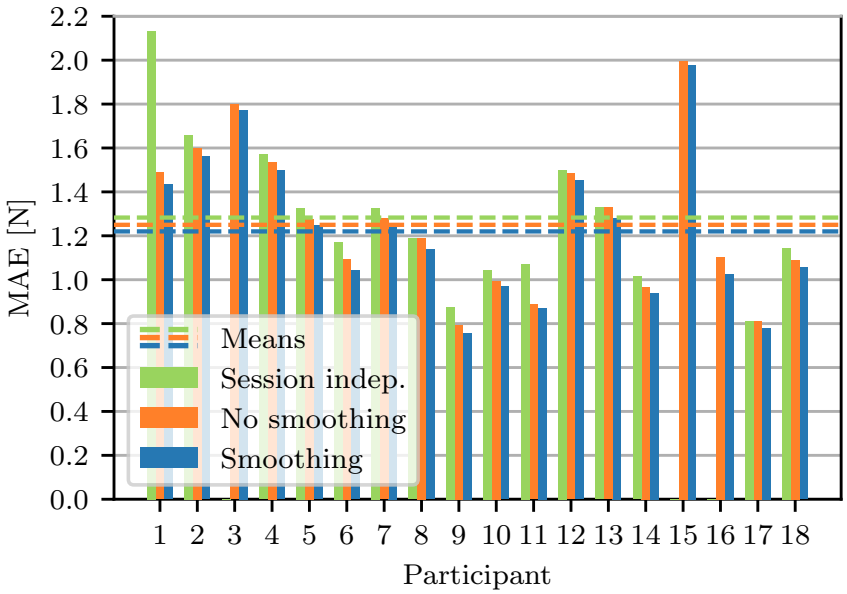


Figure 2.11: The mean absolute errors (MAE) when cross-validating linear regression models trained on the MAV and force ground truth from individual, and from multiple sessions.

be solved by using more data for training, as the original Myo gestures also work user-independently. A further practical limitation is that the Myo is not particularly comfortable to wear, especially after a longer period of time. However, we expect there to be lighter and more comfortable device developments in the future, maybe even sewn into clothing garments.

2.6 Applications

We created several demo applications employing our system. They show that it is possible to use TouchSense in real-world scenarios. A visual overview of all the applications is shown in Figure 2.1. It is important to note that for all the demos, we use the same model, i.e. did not finetune the model to the specific use case. Furthermore, in none of them, the system was calibrated during the demo session, i.e. our system can be used immediately without recording session-dependent training data. The model file is less than 140 KB large, i.e. a more general application for multiple users could easily contain several individually trained models. A video of our demos is available on YouTube [215].

The first application (cf. Figure 2.1a) demonstrates how any surface can be used as an input surface in order to control smart devices without them being required to have an interaction surface of their own. Here, a smart lamp can be controlled by pressing on any surface around it. Pressing with the forefinger decreases the brightness, pressing with the middle finger increases it. The magnitude of the change adapts to the force applied by the finger, i.e. strong presses change the brightness quickly while soft presses change it slowly.

The second application we created is a tablet app for marking text (cf. Figure 2.1b). On the screen, any finger can be used for marking the text in a semi-transparent colour. We extend the interaction surface of the tablet by an imaginary colour palette controlled by finger presses. Pressing outside of the tablet screen, e.g. on a table, controls the palette. It includes three options: Pressing with the thumb changes the colour to the next colour, pressing with the forefinger sets the stroke width according to the strength of the touch, and pressing with the middle finger reverts the last stroke. By employing this concept, we can mitigate the size limitations of the display.

We realized that our system also works for objects held in hand without further training. We took advantage of this and created a tablet drawing application using a stylus (cf. Figure 2.1c). The drawing behaviour adapts to the way the stylus is pressed with the fingers and provides a new drawing interface. When pressing with the thumb on the stylus, the colour is changed to the next colour, as before. Pressing strongly with all fingers

activates an erase function. This application shows that TouchSense cannot only enhance the interaction with smart devices such as tablets but also extend the functionality of usual "non-intelligent" objects such as the stylus.

In our last demo, we present an outdoor use case (cf. Figure 2.1d). We create an Android map application for cycling, showing a map and the current position, which can be controlled by pressing on the bicycle's handlebar without letting it go, strongly facilitating the control of the map. While riding the bicycle, the cyclist can change the map type by pressing the thumb on the handlebar, zoom out by pulling on his/her forefinger, and zoom in by pulling on his/her middle finger.

As shown with the stylus and the bicycle handlebar, TouchSense not only works on a flat surface but also on other objects. This way, our system could be used to control devices by externalizing functionality to arbitrary objects if it was possible to identify the objects through other means, such as visual recognition. For example, a mug could be responsible for controlling the volume of a sound system, a sofa for changing the TV channels, and so on.

2.7 Conclusion

In this chapter, we presented TouchSense, a mobile system to augment a human with a sensing mechanism to classify touch gestures including the thumb, the forefinger, and the middle finger and estimating the force of the touch from EMG data. Through TouchSense, all surfaces in the environment are turned into interaction areas. Besides, existing interaction paradigms can be enriched with additional features.

To gather the EMG data, we employ an inexpensive, wireless EMG armband. The EMG data is then processed on a smartphone in real time to classify the finger used in the interaction and estimate the force applied. Moreover, for training, we collect ground truth force data in order to provide labels and enable us to fit regression models for force estimation. The classification is done by a convolutional neural network we specifically designed for the purpose of EMG analysis. It runs on a standard smartphone in under 10 ms per inference and is at most 140 KB large. The evaluation showed that, as also expected from previous works, a user-independent setting is challenging, and the results are insufficient for a real application. Nevertheless, in a user-dependent, but session-independent setting our network produces satisfying results, especially when being trained and tested for only two fingers, which could be sufficient for many applications. We furthermore showcased several demos, which prove our system's real-world value.

For future improvements, one could collect more data to tackle the challenging user-independent scenario. Furthermore, one could design more sophisticated force regression methods, such as a regression network running in parallel to the classification or a unified architecture for both tasks.

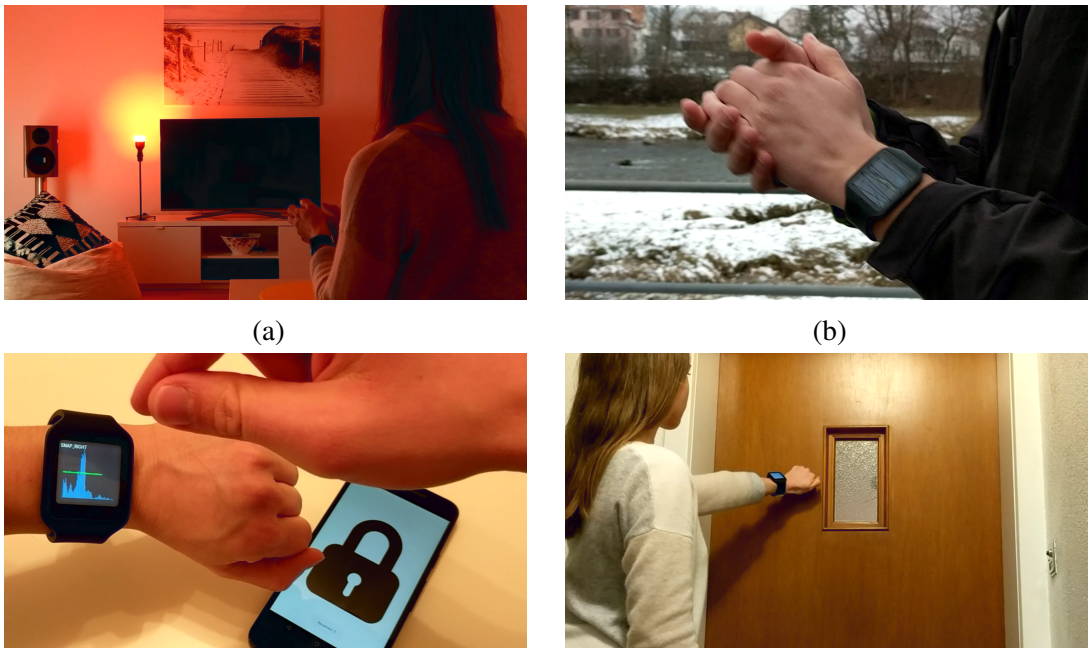
Chapter 3

Combining Audio and Motion Sensing for Gesture Recognition on Smartwatches

3.1 Introduction

Gestures form an essential part of human-to-human communication. But also in human-to-computer interactions, gestures become a viable input method, besides traditional keyboard- or touch-based input, or recent speech recognition. This is particularly true for hand gestures. In comparison to speech, gestures have the advantage of being more intuitive for some interactions and avoiding the awkwardness of speaking to no one particular [52].

A large number of different methods for hand gesture recognition have been proposed in the literature, both based on wearable as well as external sensors. In the previous chapter, we presented a gesture recognition method based on electromyography (EMG) using a wearable EMG armband. Another common approach is to use computer vision methods to classify images from a camera stream. This image-based recognition can either be performed from external or body-worn cameras, for example, mounted onto eyeglasses. The disadvantages of vision-based methods are that the hand has to be in the field of view of the camera, the camera itself is a rather expensive piece of hardware, and naturally, they cannot be used in poor lighting conditions. A prevalent line of methods use motion data from inertial measurement units (IMU) usually mounted on the wrist. Previous research has shown that this works well for several different types of gestures [161, 185, 197]. Besides the motion signals, some common gestures emit a mostly unused, but rich physical signal: sound. Gestures such as snapping or clapping, produce a distinct sound, for example,



(a) (b) (c) (d)

Figure 3.1: An overview of several applications we created around our gesture recognition method. (a) using GestEar for device control, e.g. sound or light systems; (b) music control while on the go; (c) unlocking a personal smartphone; (d) replacing the doorbell button by simply knocking at the door.

shown in the spectrogram in Figure 3.2. The features extracted from the sound signal could be beneficial for the recognition of such gestures. Recently, other researchers have explored the potential of sound-only recognition and showed that sound could be used for activity, context, and gesture classification [78, 108, 153, 192, 195].

We focus on recognizing gestures from motion and audio signals together and build a gesture set around the following gestures: snapping, clapping, and knocking. The advantage of these gestures is that they are common in most cultural regions. Snapping, for example, was already used in Ancient Greece by musicians [186]. We believe they could be used for a multitude of applications such as smart device control. Such applications have attracted notable attention in the past. Already in 1985, the "Clapper" [248], a sound-activated electrical switch, was first released (which only detects signal peaks, however, and does not recognize particular gestures). Our gesture set is further extended by additionally distinguishing which hand the gesture was performed with when snapping or knocking, and whether a clapping or knocking gesture was repeated.

Similar to the TouchSense system presented in the previous chapter, one particular goal is to achieve the recognition with minimal computational resources. Here we additionally

focus on using standard hardware without relying on specialized equipment such as an EMG armband. We develop a gesture recognition method that we can run on a smartwatch in order to create a mobile system ubiquitously available to the user. Smartwatches have the benefit that they are subject to the motion of the wrist, which captures most of a gesture's motion, and are very close to the sound source a gesture may constitute. Besides, most standard smartwatches already incorporate IMUs for activity tracking and microphones for speech recording but are naturally able to record hand movements and gesture sounds as well.

In this chapter, we present *GestEar* (gesture-ear), a method to classify sound-emitting gestures based on motion and audio together, two signals also captured by the ear. *GestEar* runs in real time solely on a smartwatch. The motion and audio data is recorded on a smartwatch and processed locally on the smartwatch, i.e. an off-the-shelf smartwatch is the only hardware element required. Other wrist-worn devices, such as fitness trackers, could be used as well if they feature an IMU and a microphone. Our main contribution and the core of our gesture classification method is a lightweight neural network architecture which takes care of both *gesture detection* and *gesture classification*. After a preprocessing step, the network receives the audio, acceleration, and gyroscope data as separate inputs, computes individual features for each, and fuses them within the network. This procedure frees us from the need of incorporating explicit sensor fusion without giving up the benefit of end-to-end training. For training our network and improving its robustness, we heavily employ data augmentation, i.e. we generate a large corpus of synthetic training data from the recorded real samples. Our network runs on an ordinary and inexpensive smartwatch with a short average inference time of under 11 ms at a high classification performance of 97% achieved in a user-independent experiment with 16 participants. Furthermore, the model file size is only 50 KB, which could easily be downloaded onto a smartwatch. We thoroughly evaluate the proposed method, also under the influence of environmental noise. We find that the particular benefit of sound lies in the detection of gestures, i.e. determining whether a segment contains a valid gesture or not, resulting in a significant decrease in the false positive rate. Based on our method, we created a working app adding fast and straightforward gesture interaction functionality to a smartwatch, which we also demonstrate in possible applications as depicted in Figure 3.1.

3.2 Related Work

There is a wide range of approaches to gesture recognition. An obvious but difficult one is to rely on external or body-worn cameras to recognize the shape of the hand visually [155]. A disadvantage is that the hand has to be in the field of view of the observing

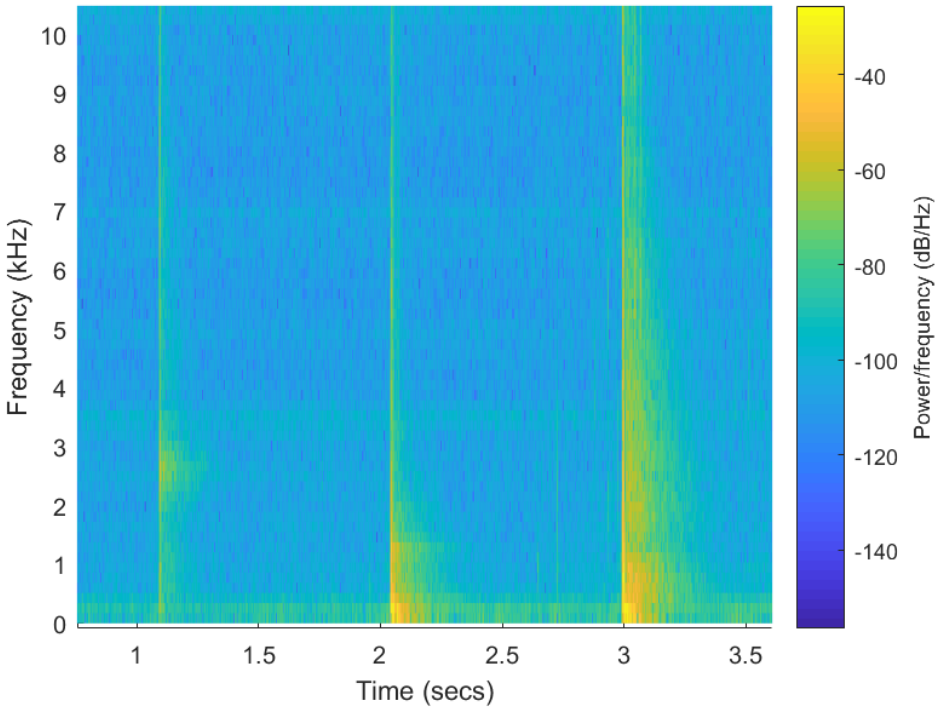


Figure 3.2: A spectrogram for snapping, knocking, and clapping (in that order). The frequency characteristics are different for each gesture.

camera. Besides, many other phenomena can be exploited for gesture recognition, such as the reflection of radar waves [177], electromyography [28], or the distortion in the electromagnetic field between a sending and a receiving antenna [113, 140]. All of these methods rely on sensors not available in nowadays common consumer devices such as smartwatches. We focus on technology that is already in use today.

Gesture Recognition from Motion Several methods use orientation and acceleration gathered from an IMU, mostly on smartwatches. For example, *Serendipity* [185] processes the recordings from the accelerometer and the gyroscope of a smartwatch to classify five finger gestures. Bâce et al. [15] employ dynamic time warping on motion sequences to distinguish eight greeting gestures based on IMU recordings from a smartwatch. Motion data has also successfully been used for touch-free target selection [161] or extending the input interface by recognizing taps around the watch [197]. Winkler et al. [187] developed a multimodal approach to recognize hand interactions with projections on a surface by visually tracking the fingers and detecting finger touches from the surface's vibrations employing a smartphone's accelerometer. These works show that it is very well possible to perform accurate gesture recognition based only on motion. The benefit of using motion

information alone is that it is less prone to environmental noise when compared to sound, but detecting gestures is difficult. We aim to show that sound can be beneficial as an additional input signal for the classification of gestures which inherently emit sound, and moreover, that recognition is possible on a smartwatch alone.

Gesture Recognition from Sound Sound has been used for many other classification problems before, most importantly, in speech recognition. Besides, it has been used in systems to help deaf people to sense their environment [116], to sense emotions and stress [106], or to perform human activity recognition [133, 190]. One option for gesture recognition is to emit inaudible tones and sense the reflection of the sound wave [77] or to inject sound chirps into the hand and measure the frequency response along the hand and wrist [196]. We do not rely on actively induced sounds, but instead, recognize gestures from the natural sounds they emit.

In recent years, several researchers proposed gesture recognition approaches that use sound directly, e.g. for recognizing handwriting [153, 192]. For hand gesture recognition, *FingerSound* [195] employs a microphone incorporated in a ring for the thumb to sense unistroke gestures of the thumb scratching against the inner side of the hand. Similar to our goal, *SoundCraft* [78] aims at classifying audio signatures from natural hand gestures such as snapping or clapping using a smartwatch. Furthermore, by incorporating four microphones, it can determine the angle of the direction the gesture was performed in, which provides an interesting interaction possibility. In both cases, the approach requires custom hardware, and the processing only runs on a computer. Laput et al. present *ViBand* [109], which uses a custom smartwatch kernel to increase the sampling rate of the accelerometer, enabling the sensing of bio-acoustic signals, which are transmitted through the body. It is not only able to reliably detect gestures in a user-dependent setting, but also to distinguish objects the user touches based on their vibration pattern. In comparison, we aim at a method which leaves the smartwatch completely unmodified. Continuing their work on sound recognition, they also present *Ubicoustics* [108], a method for human activity recognition. *Ubicoustics* segments the incoming sound signal into windows and extracts Mel-frequency cepstral coefficients from each of them. The resulting two-dimensional features (coefficients over time) are interpreted as an image and classified using a computer vision approach, a large pre-trained neural network which is fine-tuned to activity classes. This approach thereby takes advantage of robust feature detectors intensively pre-trained on a large amount of image data, which only requires an adaptation to the sound dataset. For training, audio clips of common activities from sound effect libraries are utilized, which are additionally augmented with audio from different contexts and environments. Both of these ideas inspired our augmentation method. The first difference to our work is that *Ubicoustics* focuses on recognizing human

activity and its context, e.g., car driving. Furthermore, it employs a large neural network for classification, which leads to high processing requirements. In contrast, we aim at lightweight computation besides high accuracy. Finally, we also include IMU sensor data for higher accuracy.

Gesture Recognition from Sound and Motion As we do, others have combined motion and sound sensing for gesture recognition. *TapSkin* [194] recognizes taps on up to eleven locations around the wrist when wearing a watch, based on data from the microphone, accelerometer, and gyroscope. It processes the data on a smartphone and achieves high accuracy in a user-dependent setting. In contrast to *TapSkin*, we aim at recognizing hand gestures instead of taps around the wrist. We design a lightweight neural network for classification, which reduces the amount of necessary feature engineering and enables fast inference even on a smartwatch. Ward et al. [178] use a wrist-mounted accelerometer and microphone to distinguish several wood workshop activities and combine motion and audio data for gesture detection and classification. In an offline evaluation, the authors find that the recognition using a single sensor alone works only poorly; however, when taking both into account, the performance strongly improves. In comparison, we do not rely on custom hardware, and we develop an *online* system. Nevertheless, their work provides valuable insights into the task of gesture detection, a problem we also tackle here.

3.3 The GestEar Method

Our goal is to classify nine different gesture classes: no (`null`) gesture, snap left, snap right, knock left, knock right, clap, knock left twice, knock right twice, and clap twice. The `null` class incorporates all samples which do not belong to any of the other classes, left and right refer to the hand the gesture is performed with. All gestures (except the `null` class) emit sound, which contains information about the performed gesture. For example, Figure 3.2 shows the spectrogram of an audio recording during which three different single gestures were performed. We show how this audio information can be used for gesture recognition together with data from the gyroscope and accelerometer directly on a smartwatch.

3.3.1 Data Collection

On the smartwatch, we record mono audio from the microphone at a sampling rate of 22,050 Hz and a bit depth of 16, and three-axis measurements from the gyroscope and

the accelerometer (with gravity subtraction) at a sampling rate of 200 Hz (the maximum sampling rate provided by the smartwatch we used). As the microphone samples at 22,050 Hz, we can take frequencies up to 11,025 Hz into account according to the Shannon-Nyquist theorem. To prove this is sufficient, we calculated the Fast Fourier transform (FFT) for all recorded gestures and summed up the resulting magnitudes for each frequency across all participants and gestures. The result, depicted in Figure 3.3, shows that the great majority of the signal energy lies in the frequencies below 8 kHz, hence our sampling rate is sufficient.

In order to record labelled training data, we created a companion smartphone application for the smartwatch, which displays the gesture the user has to perform, e.g. a single knock with the left hand (note that at the time of application only the smartwatch is required). Per gesture, the smartwatch records for a duration of 2 s. To collect samples for the `null` class, we let the participant perform two additional gestures: staying still and not doing anything, and 'randomly' moving the arm and hand. Consequently, our recorded gesture set contains ten gestures (eight actual gestures and two for the `null` class). In a single session, each of these ten gestures is repeated five times in random order. The smartwatch is always worn on the left wrist as watches usually are. Nevertheless, we could also train a classifier for the right hand by mirroring the motion data through the plane orthogonal to the forearm in order to transform the measurements to the new coordinate system on the right wrist (and changing the labels accordingly). The user could then select the appropriate classifier by a position setting.

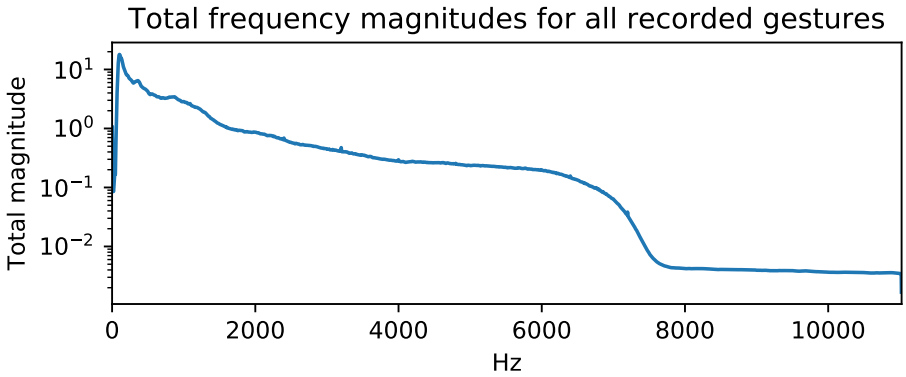


Figure 3.3: The sum of all amplitude spectra over all recorded gestures. The majority of the signal energy resides in the frequencies below 8 kHz.

3.3.2 Preprocessing

We first segment the streams of audio, gyroscope, and acceleration data into windows, and then generate features per window for each input independently. These features are then

taken as inputs by the network separately and are then combined within the network.

Segmentation We segment the incoming signals into windows of 300 ms. This window size was chosen because all of the single gestures fit into this window, and it also provides a good fit for the gestures with repetition when placing two windows after each other. Since our gesture set contains double gestures and we want to be able to classify these reliably as well, a single sample cannot only consist of a single window but must contain two of them in order to fit a double gesture. Hence, we concatenate two consecutive windows to form a *sequence*. This approach is different from simply choosing a window size of 600 ms, as we obtain two feature sets, one per window, representing the temporal structure of the input. At inference time, when we do not know the point of time a gesture is performed, we let the segmented windows overlap by half a window, i.e. the sequences overlap by 1.5 windows, as illustrated in Figure 3.4. As a result, we return a classification output from a sequence every 150 ms. This procedure ensures that a gesture fits into a sequence at inference time. As exemplified in the figure, we can take advantage of the fact that a window is part of two sequences (a sequence and the next but one), i.e. we only have to preprocess one window per sequence, except for the very first sequence.

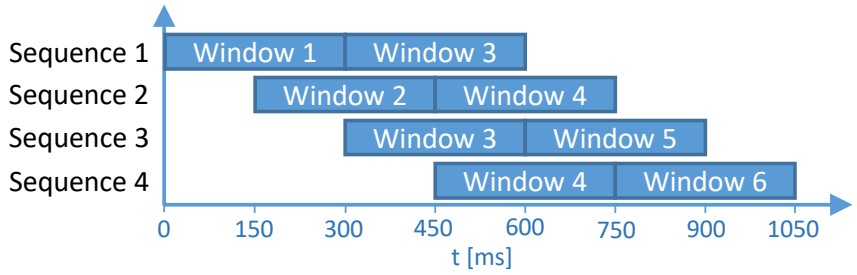


Figure 3.4: An illustration of how we segment the sequences (two consecutive windows) over time.

To segment the training data, we search for the signal's peak in each of the 2 s audio recordings and segment a window of 300 ms around the peak. The second window of the sequence is placed directly after the first, forming a valid sequence. For double gestures, which contain two peaks, we selected the first peak for the first window. We then find the corresponding window bounds in the gyroscope and acceleration recordings.

An alternative would be to classify single windows, obtain labels for single gestures, and afterwards combine consecutive labels to potential double gestures. However, we evaluated this approach and found that the recognition performance is worse, especially since consecutive windows might contain only parts of gesture signals, causing a single gesture to be recognized as a double gesture. Our approach is more robust against such mistakes, as the network learns explicitly to distinguish single and double gestures.

Preprocessing the Motion Data For the measurements from the gyroscope and the accelerometer, we do not calculate any specific features as we want the network to learn the relevant features from the motion data itself. As the single readings from the accelerometer and gyroscope usually do not arrive at exactly the same point in time, we merely resample the IMU recordings at joint time steps which correspond to a sampling rate of 200 Hz by linear interpolation. In the end, a window contains 60 ($200\,Hz * 300\,ms$) gyroscope readings and 60 accelerometer readings.

Preprocessing the Audio Data After segmentation, a single window contains 6,615 ($22,050\,Hz * 300\,ms$) audio readings. We could feed these directly to the network; however, due to the large input size, this would result in a large number of parameters. Hence, we extract features from each window's audio data by calculating the spectrum of each window using an FFT. The spectrum reveals relevant information from audio signals in a compact representation and can be calculated relatively fast even on a resource-constrained device; for some models even in hardware. Before the FFT, we apply a Hann window to avoid spectral leakage and then we zero-pad the signal to have a length of 8192 (next power of 2) so that the FFT can be performed more efficiently. Our final audio features are the magnitudes of the FFT, i.e. the amplitude spectrum. The magnitudes for the single frequencies are binned into 128 bins of equal size. We empirically evaluated that 128 bins provide a sufficient resolution for accurate classification as well as a compact representation.

As mentioned in Subsection 3.3.1, the majority of the signal energy lies below 8 kHz, i.e. the higher frequencies are less relevant and can potentially be discarded, which would help to reduce the dimensionality of the data. We evaluated keeping different numbers of bins and discarding the rest with the network we designed as described in Subsection 3.3.4. With the first 96 out of 128 bins (which corresponds to a frequency range up to 8,268 Hz), we reach the same classification performance as with the full number of bins. Hence we discard the last 32 bins. This is beneficial, as with fewer bins, there are fewer network calculations at inference time and at the same time, the classifier is less prone to high-frequent noise.

After preprocessing, a single sample is a sequence consisting of two windows, each window containing a 96-dimensional FFT magnitude vector, 60 gyroscope and 60 acceleration readings for each of the three axes (x, y, and z), i.e. the shapes of the resulting tensors are $(2, 96)$, $(2, 60, 3)$, and $(2, 60, 3)$.

3.3.3 Data Augmentation

In order to train a more robust network, we augment the data in several ways, as depicted in Figure 3.5, inspired by [108]. In the recordings, we first shift the window around the peak up to 75% of the window length and create ten sequences from a single 2 s-recording. Thereby, on the one hand, we increase the number of training samples and on the other reflect the fact that windows will most probably not perfectly fit the gesture performed at inference.

Sound signals have the beneficial property, that scaling the amplitude, as well as adding other sounds, creates a valid signal. In a second step, we create additional samples by scaling the audio signals with factors of 0.25, 0.5, and 0.75. Finally, we add environmental noise to our recordings to train the network to be more robust against such noise. We downloaded over 10 hours of freely available soundtracks from an online platform [221] recorded from many different noisy environments: city streets with people talking, from inside buildings, or even aeroplanes. For each sample generated up to this step, we produce five additional samples by adding five randomly chosen environment sound segments. We could also multiply the frequency of the audio signal by a certain factor, which would increase or decrease the speed of the audio recording. However, we do not apply this here, as we believe the speed of the gestures does not vary much, for example, one cannot clap "slowly", because then there would be no clap at all. Overall, from a single recording, we generate 240 $(10\ \text{(shifts)} * 4\ \text{(scales)} * (5\ \text{(augmentations)} + 1))$ samples.

We also require more samples belonging to the `null` class. As simply recording audio and motion data in the wild, e.g. by walking through a city, would breach the privacy of other non-involved passers-by around the participant, we instead only collected the IMU data in this manner and randomly supplemented the audio from the downloaded noise sound clips to create a large set of null samples.

3.3.4 Gesture Classification

The crucial step is the joint classification of the audio and motion sequences with the design goals of high accuracy as well as fast inference time directly on a smartwatch. Ward et al. [178] performed activity recognition based on sound and motion, employing traditional learning approaches for which they achieve an accuracy of 70%. We also first evaluated more traditional methods, an SVM and a Random Forest, for which we calculated sets of typical features from the IMU data, such as the moments and spectral features, and used the FFT features from the audio. However, we could not surpass an accuracy of 70% in a user-independent setting. Hence, attempting to reach a higher recognition performance,

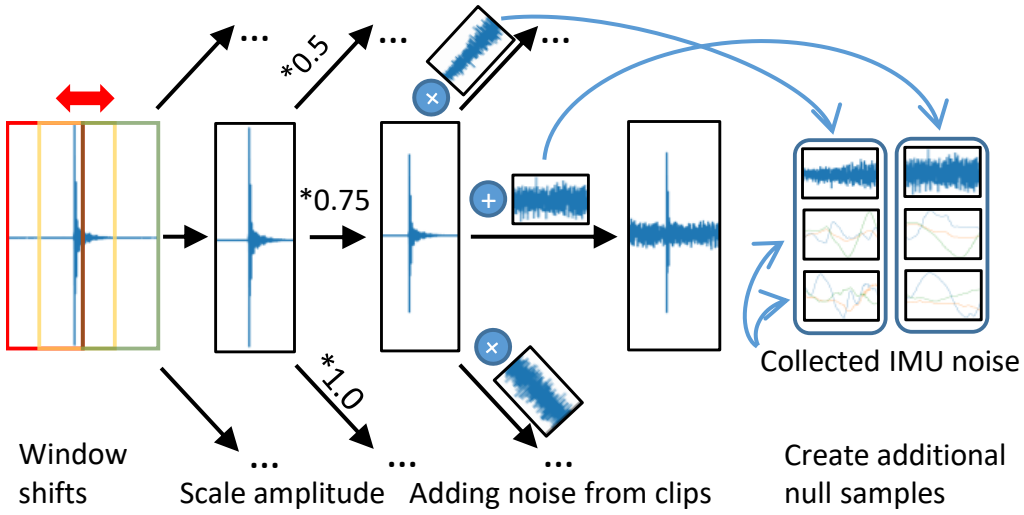


Figure 3.5: We create additional samples by shifting the segmentation window, amplifying the signal, and augmenting it with noise from sound clips. Besides, we add `null` samples by combining segments from the sound clips and IMU data from recordings of non-gesture activities.

we decided to create a neural network, which has the benefit of freeing us of feature engineering and selection, as the features are learnt within the network. We thereby can achieve an accuracy of over 97% (cf. Section 3.4). We designed the network taking into account the limited resource capabilities, i.e. it should have as few parameters as possible while achieving a high classification performance. The full architecture is depicted in Figure 3.6. An evaluation of different network designs is provided in Section 3.4.

As mentioned above, the network receives sequences of two windows each for audio, gyroscope, and acceleration. A first convolutional part, containing one-dimensional convolutional filters and max-pooling layers, extracts features from each input, followed by fully connected layers which combine these features and finally make a decision on the gesture class. The one-dimensional convolutional filters (with 16 channels and a stride of 1) detect salient features in the data, for both the FFT and the motion input, such as dominant frequency bins or certain local motion patterns. An advantage of convolutions is that they require relatively few parameters. Each convolution is followed by a max-pooling layer with a width and stride of 2, which halves the output and selects the most salient activation. We thereby obtain a very compact feature representation after this step. By repeating the convolutions and the pooling, we obtain more and more abstract features, which are also more compactly represented. These layers are applied to each of the two windows of the sequence independently. However, the weights are shared between the operations, i.e. the same features are extracted from both windows. Note that we do not interpret the two windows as an image and do not perform two-dimensional convolutions

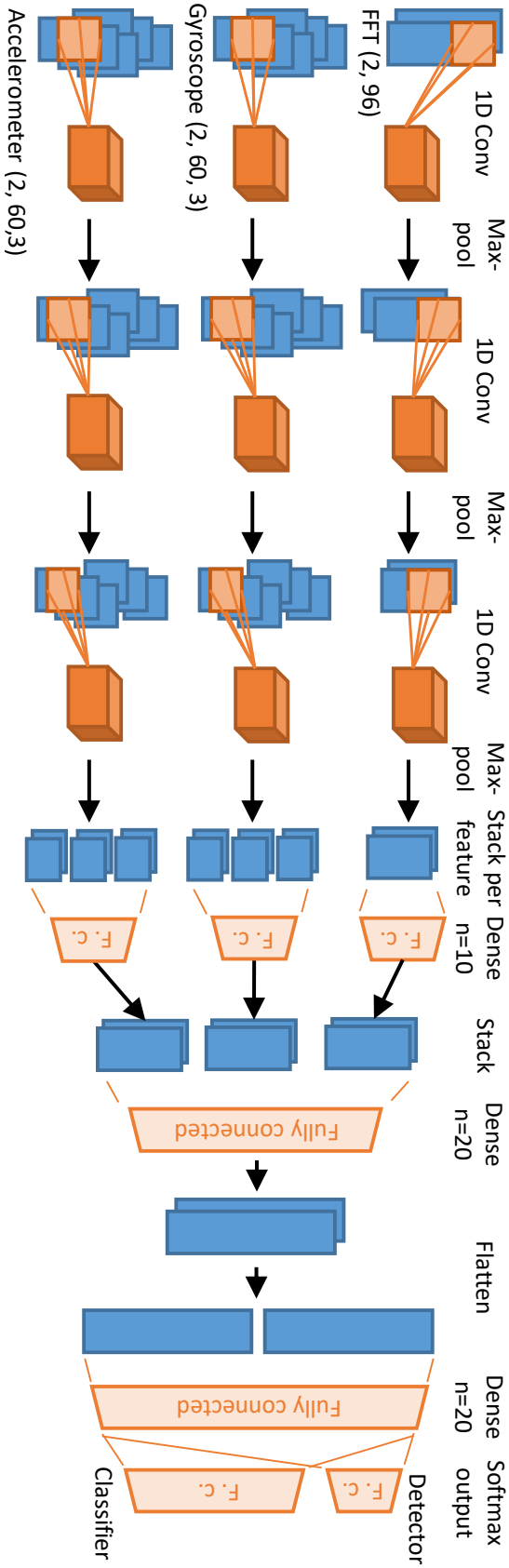


Figure 3.6: The neural network we designed for gesture recognition. The input are sequences containing two windows of audio spectra, and time-domain gyroscope and accelerometer data. The network consists of a convolutional part for extracting features from each input type separately, followed by fully connected layers which combine these features (F.c. stands for fully connected). It is responsible for both gesture detection and classification by providing two outputs.

across the windows, as done in [108].

Afterwards, we add a fully connected layer (10 units) to each of the three convolutional parts applied to audio, gyroscope, and acceleration input in order to combine the learnt features and create a final individual representation of that input. These representations are then stacked and fed to another fully connected layer (20 units) which now takes all features into account. Up to this step, all operations are still applied to each of the two sequence components separately, but with weight-sharing. Finally, we flatten the tensors along the temporal axis and apply another fully connected layer, which evaluates the temporal relationship. At this point, we could include a recurrent layer like an LSTM (Long Short Term Memory). However, an LSTM requires a high number of parameters relative to the little number of two windows handled here. Thus, we decided to follow the simpler version of flattening the tensors and employing a fully connected layer. The final classification takes place in a softmax fully connected layer. We use ReLU (rectified linear unit) as the activation function in every layer except the last one.

3.3.5 Gesture Detection

In order to detect whether a sequence contains a gesture, we can first check if there is significant energy in the sound signal. We only submit the sequence for classification if the sum of the FFT magnitudes in the sequence is greater or equal to 0.15 (arbitrary unit after FFT). Otherwise, we drop the sequence. We determined this threshold by averaging all magnitudes from valid gesture sequences obtained in the collection trial, which results in an average magnitude of 0.591 and choosing the threshold to be roughly 25% of that. However, in the first experiments, we found that our network and this threshold work well in case the sequences were valid gestures, or there was little audio noise. Nevertheless, we also obtained many false positives from environmental noise which surpassed the threshold criterion. To counter this, we introduce a gesture/no gesture-classifier besides the classification of the gesture type, which determines whether a sequence represents a valid gesture or not. In previous works, this is a separate step, for example, using a binary SVM, which increases the processing time for a single sample as it has to run through two classifiers. On the contrary, if we regard our network as a feature detector (the part up to the last layer), the extracted features would also be useful for the gesture detection classifier and it seems an unnecessary loss of resources to compute additional features for it in a separate detection process. Hence, we incorporate gesture detection in the neural network by adding a softmax output layer with two units, which uses the same network body, as shown in Figure 3.6. The result of the classification at inference time is only accepted if the detection is positive.

Another advantage of incorporating gesture detection is that we can use a pretraining scheme. The data available for training are the set from the collection trials, which contains roughly the same number of samples for each gesture class and the supplementary set of `null` samples resulting from the sound clip audio data and additionally collected motion samples. If we were to train only the gesture classification, we would have to balance the dataset to have an equal number of samples for each class, leaving many of the additional `null` samples unused. However, if we first train the gesture detector, we balance all `null` samples and all valid gestures. Thereby, we obtain an accurate gesture detector and at the same time, a network body which has already seen many `null` samples and valid gestures and thereby learnt relevant features for gesture recognition. This network body now serves as initialization for training the gesture classifier. We now balance the dataset for all nine classes. To maintain the gesture detection performance, we jointly train both outputs during this phase by backpropagating the errors from both outputs through the network.

For both training phases, we use an Adam optimizer [101] with a learning rate of 0.001, a batch size of 100, and we control for overfitting. For the gesture detection, we train for five epochs. For the second step, we train for 15 epochs.

In total, our network only has 9,989 parameters, which enables a fast execution. In comparison, *Ubicoustics* [108] uses a deep convolutional network (VGG16 architecture [154]) with over 138 million parameters, which consequently takes significantly more computation resources.

3.3.6 Post-processing at Inference Time

At inference time, the sequences overlap by 1.5 windows to increase the chance that a gesture falls into a sequence completely. For a double gesture, e.g. clapping twice, the first clap could lie in the second window of a first and in both windows of the following sequence resulting in the output of “Clap” for the first and the expected “Double clap” for the second sequence. To avoid this, we defer the output of a gesture until three more (overlapping) sequences are classified and select the gesture with the highest average probability (from the softmax layer) over the four votes.

3.3.7 Implementation

The smartwatch used throughout this work is a Sony Smartwatch 3, an inexpensive model with Wear OS. At the time of writing, the release date of this smartwatch lies over four

years in the past; hence we demonstrate that running our method does not require the latest hardware. We implemented the processing pipeline in Python and a corresponding pipeline in Java to run on the Android smartwatch. To perform the FFT on the smartwatch, we use the Noise library [207]. The neural network is implemented in TensorFlow [242], trained on a desktop, converted to the TFLite model format, and transferred to the smartwatch. The model file size is only 50 KB, so it can easily be shipped within a mobile application. The code is available on GitHub [209].

3.4 Evaluation

For training and testing our system, we collected data from 16 participants (20 to 55 years old, 25.75 years old on average, five females, four left-handed). In a single session, we recorded each gesture five times in a randomized order. For each participant, we recorded five sessions.

3.4.1 Recognition Performance

In a first experiment using all data to perform 10-fold cross-validation, we achieve a high average accuracy of 96.9% (F1 score: 96.9%), which shows that our network is able to capture the characteristics of the input signals. In a second experiment, we attempt the most challenging setting, cross-validation on participants. We exclude a single participant's data from the training set and test on this data. We repeat this for each participant. Here, we achieve an average accuracy of 97.2% (F1 score: 97.2%), which proves that our classifier also generalizes across users and implies that it has to be trained only once and can consequently be used by anyone. For all participants, we achieve an accuracy above 95%. The confusion plot is given in Figure 3.7. Most errors happen for similar gestures, such as "Snap right" and "Knock right". These two are harder to distinguish from another than their left counterparts, as there is no IMU data available for them.

3.4.2 Comparison of Network Architectures

We evaluated different network designs in order to justify the choice of our network (training and testing the same way as before). The results, together with the total number of parameters for each network, are visualized in Figure 3.8, denoted by the abbreviations used in the following. Table 3.1 specifies the different models in more detail.

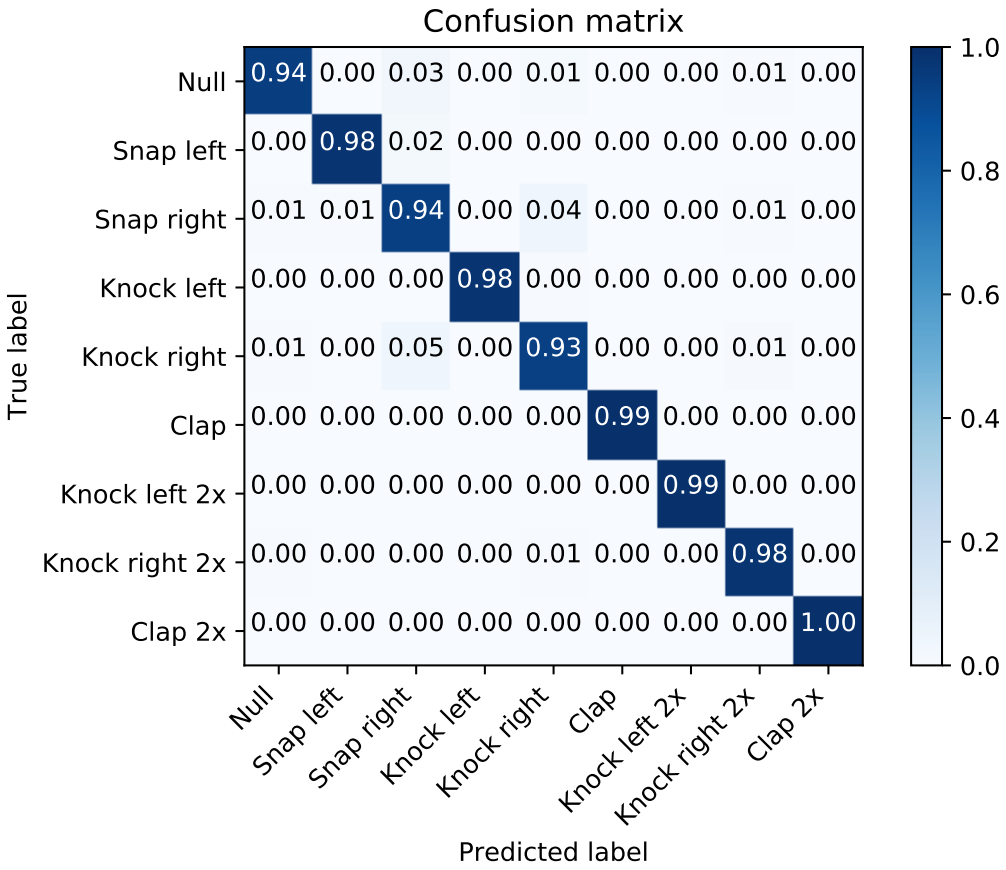


Figure 3.7: The normalized confusion plot for the cross-validation across participants.

First, we create variations of our default network by changing its width, i.e. multiplying the number of units in each layer by a factor ranging from 0.25 to 1.5 (denoted by `w=factor`). For convolutional layers, this means to scale the number of filters, for fully connected layers to scale the number of units (we round down any non-integer). We also change the depth of the network, e.g. by reducing the number of convolutional levels (a one-dimensional convolution followed by a max-pooling layer) or altering the number of dense layers. Note that with less convolutional levels, the number of parameters increases, as the internal representation after the convolutions will be larger. Thus the following fully connected layers will have more weights (`wdl`: without last dense layer, `adl`: additional dense layer, `c=n`: number of convolutional levels). We believe that among all these variations, our default network (`w=1`) offers a good trade-off in terms of accuracy and the number of parameters, i.e. runtime.

For comparison, we also trained a simple dense network (`d`) consisting only of two fully connected layers with a similar number of parameters as our default network. Our network shows a higher performance when compared to this simple network. We also implemented a recently popular mechanism, attention, proposed by Bahdanau et al. [16]

for machine translation. The term encompasses a range of mechanisms, which learn to emphasize a certain input, both on a feature (spatial) level or in a temporal context. First, we replace the last fully connected layer in our network by a spatial attention module (`sa`), which learns to generate a weight vector from the input and element-wise multiplies the weight vector and the input. Second, we apply a temporal attention mechanism (`ta`), which replaces the flattening of the sequence tensors and learns a weighted combination of the sequence tensors. Finally, we combine both parts (`sta`). However, neither of these approaches showed any benefit.

3.4.3 Importance of the Audio Features (Ablation Study)

Since previous works have shown that gesture recognition from motion alone works well, one might pose the question of how much the sound input actually contributes. First of all, it enables the recognition of the purely right-handed gestures, i.e. for three out of nine gesture classes, as there is no motion information available for these. Nevertheless, the question still remains concerning the six classes including the left hand (and the `null` class). The following experiments all apply only to this reduced set of six left-hand classes. The accuracy for our network including all features on this subset of gestures in the user-independent setting is 98.1%. We remove the audio input (and all directly following layers) and rerun this experiment (including training). Remarkably,

Table 3.1: The different network designs in our comparison. The specifiers relate to the abbreviations used in Figure 3.8.

Specifier	Description
w=0.25	Default network, width=0.25
w=0.5	Default network, width=0.5
w=0.75	Default network, width=0.75
w=1	Default network
w=1.25	Default network, width=1.25
w=1.5	Default network, width=1.5
d	Only dense network part on concatenated input
c=1	Default network with one convolutional level
c=2	Default network with two convolutional levels
wdl	Default network without the last dense layer
adl	Default net with an additional dense layer (20 units)
sa	Default network with spatial attention
ta	Default network with temporal attention
sta	Default network with both attention mechanisms

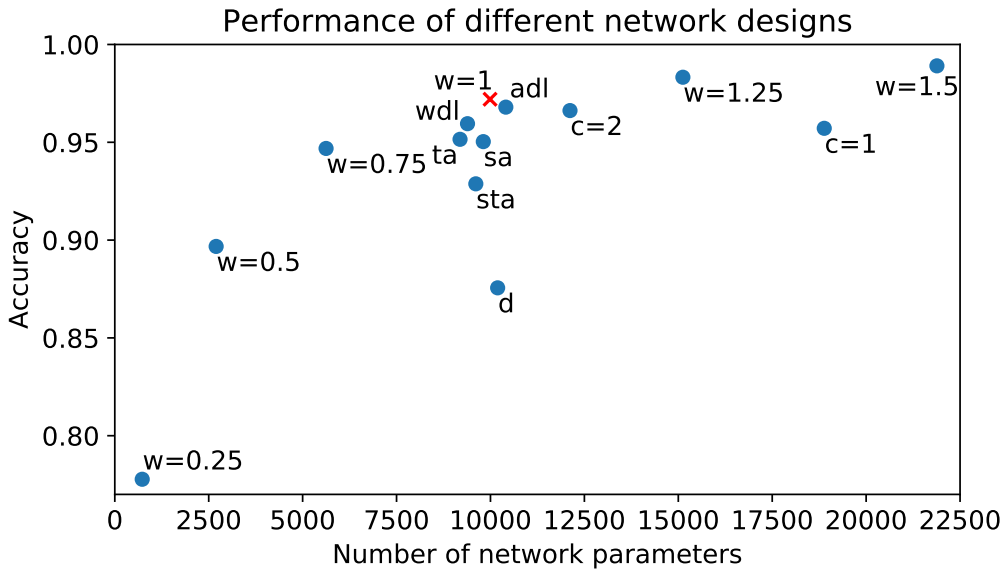


Figure 3.8: The accuracy of different networks w.r.t. to their number of parameters, labelled with the abbreviation of the design. Our default network is marked by the red cross.

the accuracy is only marginally worse (97.8%). However, up to this point, we classified the preprocessed well-segmented samples (cf. Subsection 3.3.2). In a more practical and also more challenging setting, we would run our method on a continuous stream of overlapping sequences, most likely containing many more sequences from the `null` class. Therefore, in a new experiment, we replicate our application pipeline and use our full 2 s recordings gathered during data collection as real-world stream samples instead of only employing the segments containing the gestures. As for the real application, we continuously segment the 2 s recordings from the beginning to the end and classify the resulting stream of sequences (including post-processing). The ideal outcome would be a single, correct label for each recording (except for `null` recordings). As before, we train and test in a leave-one-participant-out manner. We measure the recall (was the correct label predicted?) and the false positive rate (how many incorrect or additional "correct" labels were delivered?). Including sound and motion as features, we achieve a high average recall of 94.4% and a low average false positive rate of 5.5%. When training and testing without sound, the recall is still high (87.8%), however, the false positive rate rises to 45.5%, i.e. the detection of gestures is much more reliable with sound.

3.4.4 Performance in Noisy Environments

To evaluate our method's performance not only with the audio noise which we added to the training data but also with real noise, we recorded five additional sessions for five of

our participants while playing white Gaussian audio noise at increasing noise levels. The decibel levels resemble noise ranging from little noise to louder traffic. We train on all the other data as usual and test on the noisy sessions. For these sessions, we did not add any noise augmentations during preprocessing to not further alter the audio noise. The accuracy decreases with increasing environmental noise (cf. Table 3.2). This, in particular, affects the right-handed gestures, e.g. "Snap right", as these cannot be recognized from motion. For louder noise, they are more often assigned to the `null` class, as the recordings then resemble `null` samples with noise augmentation used for training. Nevertheless, the classifier proves to be relatively robust against noise.

Table 3.2: Testing on different levels of environmental noise.

Noise level [dB]	50	55	60	65	70
Avg. accuracy	94.1%	95.6%	94.7%	91.1%	91.2%

3.4.5 Runtime

We also investigate the time requirements for running our method on the smartwatch, including the preprocessing steps as well as running the network for inference. We let the app process 1,000 sequences and calculate the average runtime per sequence. Note that the network is only run if the sum of the FFT magnitudes in a sequence is high enough, i.e. only for a fraction of the recorded sequences. For this experiment, we removed the magnitude threshold so that every sequence is classified to calculate the correct average. Overall, processing a single sequence (600 ms) takes 64.08 ms. The shares of the runtime for the specific processing steps are depicted in Figure 3.9. The effort for running the network inference is low at 10.56 ms on average. The greatest effort is spent on calculating the FFT (43.47 ms). Performing the resampling of the motion data takes 10.05 ms. Note that the two preprocessing steps and the inference are pipelined. We believe that both network inference and FFT could be accelerated by dedicated hardware in the future. Nevertheless, this evaluation shows that regular use is possible even without further optimization.

3.5 Applications

We believe our method provides a fast, simple, and intuitive way to interact with the smartwatch itself, but also with other devices. We created several applications to demonstrate GestEar in real-world settings. Figure 3.1 shows an overview of all demos (we also published a video on YouTube [210]). We always used the same network model, which

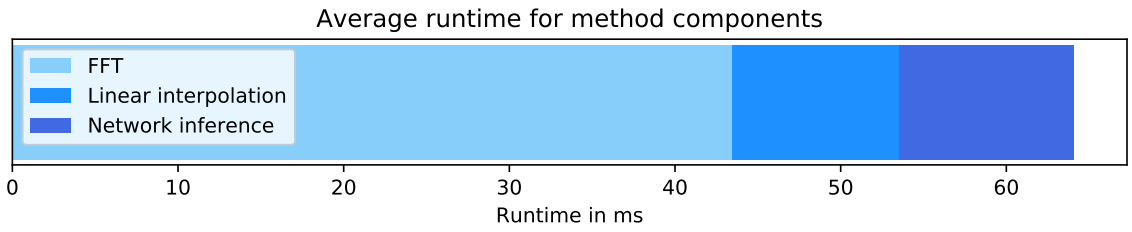


Figure 3.9: A plot showing how the different components of our algorithm contribute to the total runtime.

we did not tune to any specific user or application. The first demo shows device control in an environment consisting of a sound system and a smart lamp. We show how the user can give commands easily and quickly, such as turning the music on and off, changing tracks, turning the lamp on and off, and changing its colour. We believe that there are also many other devices which could be controlled this way, such as the television or even cleaning robots; the two devices here should only highlight the potential. We extended this scenario to a mobile setting, for controlling the music on the smartphone while walking or running. This could also be used for accepting or declining incoming calls.

In the second application, we take advantage of the fact that some of the gestures cannot be reproduced by people other than the user him-/herself. For example, the "Snap left" gesture is only recognized if the corresponding motion of the wrist also occurs. Potential adversaries cannot provide that. We can hence use "Snap left" as a command to unlock a companion smartphone in an easy and simple manner.

Finally, we believe that GestEar can enable novel interactions. We created a demo that shows how our system could replace the button of a doorbell. When a user knocks at the door, a message is sent to the flat's home control system, which then rings the bell, without the user ever having to touch the door bell button itself.

3.6 Conclusion

In this chapter, we presented GestEar, a fast and robust method for real-time recognition of eight sound-emitting gestures using the microphone and motion sensors of an unmodified smartwatch. We designed an efficient neural network to jointly perform gesture detection and recognition; we showed how to train the network for both of these tasks, and how to create a large training dataset from a relatively small number of recordings. Besides different gesture types, we were also able to distinguish whether the gesture was performed with the right or the left hand, or whether it was performed twice in quick succession to

form a double gesture. The resulting model is lightweight; it has a size of only 50 KB and requires only 10.6 ms for inference on average directly on the smartwatch. Our method achieves a high accuracy of 97.2% in a user-independent setting with 16 participants, indicating that the method would only have to be trained once to work even for unknown users. GestEar demonstrates the advantage of combining readily-available sound and motion input on smartwatches for enhanced gesture recognition. We also created a working app enabling fast and simple gestural interactions, which could easily be added to existing smartwatches as a software update, and showed several smart home applications employing these interactions.

Chapter 4

Connecting and Controlling Appliances through Wearable Augmented Reality

4.1 Introduction

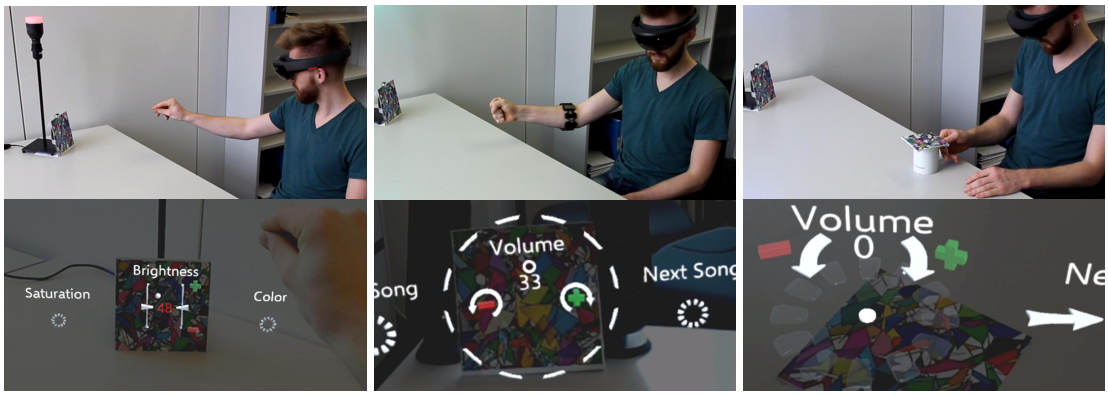
The number of interconnected devices around us is continuously growing, and it may become challenging for users to control all these devices when control interfaces are distributed over mechanical elements, apps, and configuration webpages. If devices are even supposed to be connected and work together, this challenge is intensified. As mentioned in Chapter 1, new ways of interaction between humans and appliances have gained significant interest with the emergence of networked smart appliances. Since the times appliances could be operated only by hard-wired knobs and buttons, we have witnessed the inventions of a number of convenience features. Remote controls were introduced in the late 1930s, allowing to comfortably control a single appliance from a distance. Since then, we reached the point that home systems allow almost universal device control via smartphone apps or partly even via speech and gesture recognition systems. This process can be described by the term *user interface externalization*. The physical user interface of an appliance is *externalized* from the appliance to the remote control, to a companion smartphone app, or in an abstract sense to the speech or gesture command language.

As appliances in our homes are becoming smarter with improved sensing, actuation, and computing capabilities as well as connectivity to other devices and cloud services [122], there are new opportunities for more advanced interaction and also for externalizing the appliances' user interface to other devices. Besides the ability to control things remotely,

further advantages of such "outsourced" user interfaces are smaller cost, overcoming the lack of display space, and simpler UI feature updates via software.

In this thesis, we present different concepts of externalizing user interfaces of appliances to personal wearable computers. More generally, we show different concepts of how a network of smart things worn on our body allows universal interaction with a network of smart things in our environment. In the two preceding chapters, we focused on gestural interface methods, and the following chapter is about tangible user interfaces the users can create themselves. In this chapter, we employ visual representations of device characteristics and actions in augmented reality (AR) and allow the user to interact with the devices through this AR interface. For example, a digital colour lamp may be represented by its brightness and colour values, which can be manipulated via one-dimensional sliders. As we make use of a head-mounted AR display, all representations appear in the user's egocentric view, i.e. allow the direct interaction with an appliance, without an intermediary remote control device. Furthermore, we can create one unified interface for all devices and, in particular, the devices themselves are not required to have a display or exhibit any kind of direct input and output.

This concept holds great potential for the future, especially as AR headsets will become



(1) (2) (3)

Figure 4.1: We present a concept and a prototype system for universal control of networked smart appliances. The appliances externalize their user interfaces to networked wearable computers. More specifically, a head-mounted display renders automatically generated digital user interface primitives overlaying the physical objects. We compare three ways of manipulating the primitives: (1) a pinching gesture, (2) other natural gestures like arm rotations, and (3) utilizing tangible physical objects at hand. The images in the top row show the outside view while the images in the bottom row show the AR view at the same moment in time during the different modes of interaction.

smaller, and we believe it is necessary to study the user's behaviour in such an environment as well as the different possible interaction modalities. For this reason, we built a working prototype of the described concept with a Microsoft HoloLens for displaying the visual AR representation, and various techniques for input. Devices are recognized via visual markers, and the user can select a specific smart thing with his/her gaze, i.e. by looking at them. Then, the supported interaction primitives (knobs, sliders, etc.) overlay the devices in the AR view. In the spatially registered AR view, the visual content describes the appliance's capabilities, and the interaction primitives are represented depending on how they can be manipulated.

For actuating the smart objects, i.e. providing input to modify the primitives and thereby the device parameters, we design three different possibilities which we enable by employing different wearable computers, e.g. a Thalmic Lab's Myo or the HoloLens itself: first, using a simple pinching gesture to adjust virtual sliders and click on buttons (Figure 4.1(1)), second, using a larger gesture set recognized by an electromyography armband (Figure 4.1(2)), and third, by displaying the representations on a physical object, which allows controlling an appliance by moving or rotating the physical object (Figure 4.1(3)). The third option can be seen as an extreme form of externalization, which even allows embedding "dumb", unconnected objects in the space of smart, connected appliances. We compare the three possibilities as interaction scenarios in a study with 25 participants.

Besides the one-to-one control relationship between a user and an appliance, many relationships between devices are only configured by a human user, and the further interaction between the devices may be automatic. For example, a light sensor might be connected to a smart light, thereby controlling its brightness. The concept described above does not reflect these (potential many-to-many) relationships as it only allows the interaction with a single device at a time. In comparison to configuring single devices alone, it is usually considerably more cumbersome to connect devices to work together, as the control of the devices is distributed over several interfaces. Often, these connections are simply fixed, for example, there usually is a dedicated remote control for the television, another for the sound system, and the connection between switches and specific lights are hard-wired. Being able to configure these connections dynamically has the great benefit that devices can be reused for different purposes, and the collective functions can be changed and set to one's current preferences. With a steadily growing number of devices, the management of the connections is an important problem to consider, as there is a quadratic number of potential connections between devices.

A solution which allows the simple configuration of device connections would not only make it easier for users to understand and manipulate combined devices but also set all dedicated controllers, such as remote controls and switches, free to be used to control potentially any smart device. Therefore, we build upon our work on the control of

single appliances in AR and present *ConnectAR*, an approach to visualize and configure connections of appliances and controllers in augmented reality using a head-mounted AR display as depicted in Figure 4.2. Devices and controllers are selected by looking at them and then provide a virtual button the user can click on employing the simple pinching gesture we found as the best interaction method in the first part to select them for connection. Consequently, the counterpart is selected in the same way. We then show the connection as a virtual line in the 3D space and at the same time, create a logical control connection between the controller and the device. Henceforth, the user can employ the configured controller and controlled device independently of our AR system, i.e. the head-mounted display (HMD) only has to be worn when the connections are to be checked or configured. The only requirement is that all devices are connected to a server via Wi-Fi. By using the HMD and a wireless connection, our system is mobile and spatially unlimited; devices could even be located in different rooms.

4.2 Related Work

A recurring term in our work is the aforementioned externalisation of a user interface. Here we externalize interfaces to an AR representation. One way of externalization deployed nowadays is the use of handheld devices such as smartphones or tablets to

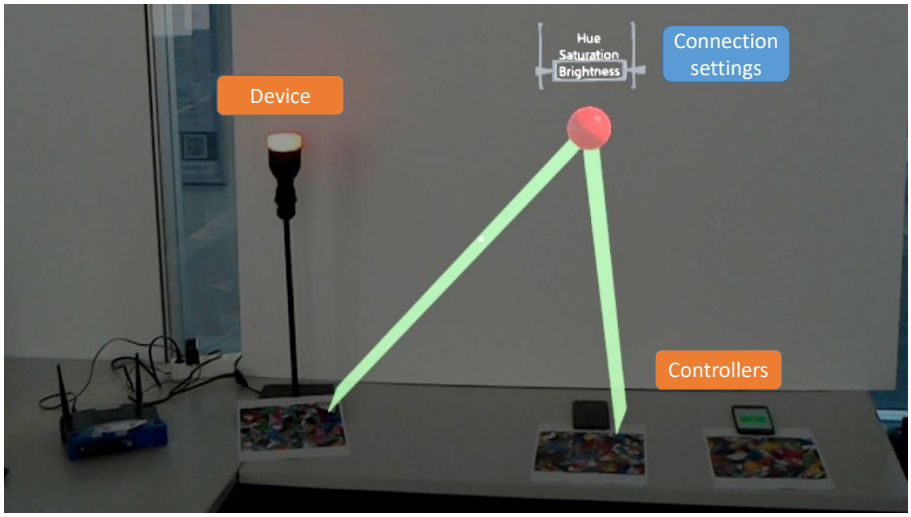


Figure 4.2: We present a second prototype for fast and simple configuration of device connections, such as between a common controller and a light, using simple gestures. The connections are visualized in augmented reality making it easy for the user to understand existing connections and to configure them. For simplicity, we use Android smartphones running an app as controllers.

display information and controls of other devices. This is also a common solution for many commercial products. Interesting in the scope of our work are approaches which also facilitate the selection of a device. Several researchers explored this by combining device recognition and device control. Mostly, devices are recognized visually from camera images [53, 124], but also other modalities such as visual codes [102] or electromagnetic signatures are used [189]. Some works additionally take the user's location, i.e. the proximity to the devices, into account for the selection process [53, 111]. Subsequently, user interfaces on the handhelds provide visual control elements to actuate the devices. While these works approach the same problem as we do, they require the user to interact with the handheld, i.e. a proxy device in between. In contrast, we provide a direct interaction interface in the AR space observed from the user's egocentric perspective.

Improving on the works mentioned above, several researchers employ visual AR on handhelds for device control [82, 90]. This allows the user to interact with the visual counterpart of the real device in the camera image after the device has been recognized and not only with a simple UI representation consisting of buttons and sliders. Huo et al. [86] enhanced this method in *Scenariot* by locating the devices through ultra-wide-band RF units, so that the application always knows the position of the devices even if they are outside of the camera's view. Nevertheless, the user still requires a handheld and merely interacts with a visual representation of the device. In our approach, the user can directly interact in the mixed reality space "on the device" itself without having to look back and forth from the remote control to the device. *PICOntrol* [152] is also based on a handheld device but uses an incorporated projector to display a graphical interface on the controllable appliances and employs visible light communication to transmit control information. Actions are selected using buttons on the handheld and by the transmission of the corresponding light patterns. Thereby, the visual UI elements augment the real device and not a virtual counterpart. Nonetheless, the user still has to hold a handheld device.

Finally, only a few researchers have used head-mounted displays for interaction in the AR space. Sorgalla et al. [157] recently proposed a system called *ARGI*, which uses the HoloLens to augment the user's view with interaction widgets placed on controllable devices. While the basic concept is similar to ours, the interaction possibilities do not go beyond the standard pinching gestures provided by the HoloLens. Going further, we create three different ways of interacting with the devices and conduct a user study to investigate the use of the AR system and the interaction methods. Another system, *Ubii* [115] uses a Google Glass to represent digital devices such as computers or smartphones and the files they contain in the glass' view and supports two main operations through simple pinching and dragging gestures: the transfer of files between two devices or to a device such as a printer and the pairing of devices. *Ubii* is different from our system by only targeting digital devices and supporting only two specific operations for these and not allowing

device control in a general sense. *AmbiGaze* [167] employs a wearable eye tracker to allow device interaction using a target selection method called pursuits, which correlates eye movements to the movements of targets displayed on the device (cf. Subsection 1.1.5). By following the target, the user can select certain actions. However, this only works if the device has a display to show targets or exhibits mechanical movement itself. All other devices require a stationary projector displaying the targets, i.e. in comparison to our concept, it requires an additional augmentation of the devices.

Further interesting in the scope of our effort are works which externalize interfaces to physical elements, which were not deliberately designed to act as a controller. One possibility to do this is to reuse existing controls, e.g. use the switch of a light to play or pause a sound system [37, 83], as we expand upon below. On the other hand, one could use simple physical objects available in the environment, as we intend to do. This was already proposed by Ishii et al. [87] more than two decades ago. Recently, Pohl et al. [138] explored the space of everyday objects potentially useful for control and found that there is both a diverse set of objects available and potential use for them. Henderson et al. [79] proposed so-called *Opportunistic Controls*, a system to support the use of tangible elements, e.g. bolts of a machine, for input when using a head-mounted AR display. They found that the opportunistic controls support faster completion times than a baseline technique, which uses simple virtual buttons in AR. Several previous approaches which allow using simple objects as control elements, instrument a space which is observed by cameras from above or below [11, 33, 43, 71]. The cameras can recognize the movement of the objects within the scene. Some also use projectors to display information on the objects. The concept of these works is very similar to one of our approaches. We, however, build a system which only requires a HoloLens instead of an instrumented workspace; hence, our approach is mobile and potentially unbounded.

We further discuss works concerning the configuration of appliances, as already shortly highlighted above. Heun et al. [82, 83] present the *Reality Editor*, a tablet AR application allowing the configuration of controls and devices. Both the controllers and the objects are recognized by the tablet camera through the use of visual markers. After recognition, an AR overlay is shown. Subsequently, the user can not only control and configure the discovered appliances one by one but also simply draw a line in between them to connect controls and devices in the field of view of the camera. The device can then be operated via the connected controls. This is similar to what we envision for the configuration of devices: visualizing the connections by lines shown in AR and allowing their simple modification. Nevertheless, the *Reality Editor* is still limited to the tablet screen, and the user is not interacting with the devices themselves but with the tablet, a proxy object acting as a remote control. We intend to overcome this limitation by using a head-mounted AR display and the spatial mapping of the environment space. This also allows the user to

directly interact with the physical devices themselves from his/her egocentric point of view. Moreover, our approach makes it easy to configure devices in a large space.

Instead of employing a tablet, the *InSight* system [37] dynamically links two devices based on the user's gaze. It uses an eyewear-mounted infra-red (IR) laser emitter and IR sensors attached to the smart devices. This makes it possible to detect the devices the user's gaze falls upon. The user can look at an input device and an actuator device to establish a link between the two.

Park et al. [135] present a prototype system to define inter-device rules in a head-mounted AR display using gestures. However, their system applies only to abstract virtual devices and not to real physical devices.

Kubitza et al. [103, 104] approach the connection of smart devices by providing a physical hub which the devices connect to wirelessly and through which the user can set up device rules. They offer an API to a wide range of devices and allow their configuration with Java script in a website providing unified access to the devices and their sensors. While this significantly simplifies setting up an environment of smart devices for expert users, it is questionable how easy this is for inexperienced users. We provide an easily understandable method by allowing the user to view and modify the connections of devices in AR by using simple gestures.

The idea to visualize connected devices has already been proposed in previous work. Mayer et al. [123] create a so-called magic lens, a tablet AR application to show web communication links between smart devices. A software logging agent running on each device collects the connection information sent between the devices and forwards this information to a back-end server which provides the information to the tablet application. The devices are recognized by the tablet camera, and the camera preview is overlaid with the virtual connections. We build upon this idea; however, we intend not to focus on pure information exchange between devices, but visualizing control links and also allowing the user to manipulate these links. Moreover, by using a head-mounted AR display, we free the view from the limited screen size of a tablet.

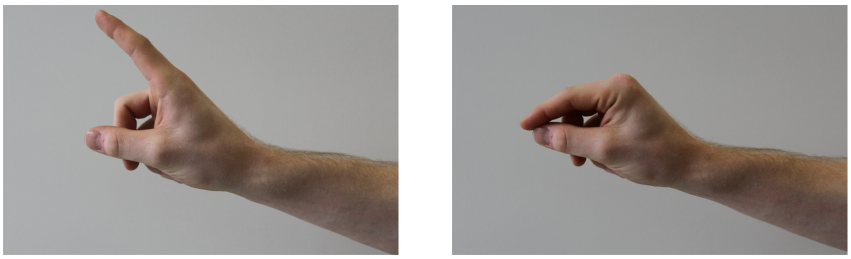
4.3 Universal Appliance Control through Wearable Augmented Reality

We propose a concept to enable direct interaction with appliances through head-mounted AR. As mentioned in Section 4.1, we represent the devices by widget representations in the egocentric AR view. We designed and implemented three different ways to provide

input, interaction scenarios, which are described in the following, and compare these scenarios in a user study. Our general interaction paradigm is to visually augment an appliance with a representation of its parameters and the way to apply modifications to them (see the bottom row of Figure 4.1). We recognize the devices with an egocentric camera and by their individual visual markers. The user sees the point of his/her gaze in AR, which is approximated by the head pose and can select an appliance by looking at it, more precisely at its marker. This procedure is the same for all three scenarios. However, each scenario adopts a different type of user input to change the parameters of a connected appliance and thus also incorporates a different state visualization of the appliance. Our intention was to find out how users would interact in the AR space. Hence we only choose scenarios within this space. Since visual AR is able to show interaction elements, we allow direct interaction with these elements. We want to compare this to the use of natural gestures. Finally, we also include a tangible object. We do not include a control device like a connected knob, as this would defy the idea of a wearable system that can always be with us. We furthermore do not include speech commands, as we use visual elements to represent the appliances, which provide the ability to not only show state information but also allow their manipulation by moving and clicking them. In terms of this representation, it appears more reasonable to manipulate the elements and their underlying states using gestures than speech, similar to how we use gestures for manipulation with ordinary computer environments such as with a mouse for a desktop, or a touchpad or screen for tablets and laptops. Moreover, speech interaction with appliances is known to have acceptance problems due to social awkwardness, especially in public [52]. In the following, we describe the three scenarios in more detail. A demonstration of each scenario is included in a demo video we created [211].

4.3.1 Interaction Scenarios

Scenario 1 In the first scenario, we aim at a simple mode of control, composed of simple gestures and simple widgets, which allows for direct interaction with the devices, i.e. the user will also look at the devices while interacting because the widgets are placed on the devices themselves. We use the predefined HoloLens gesture, pinching thumb and forefinger as depicted in Figure 4.3, as the only option for user input. The gesture is recognized by the HoloLens based on visual recognition. It can be used to click on an AR element by simply tapping as well as dragging an element by holding the closed form of the gesture, as shown in Figure 4.1(1). This enables us to use virtual sliders and buttons as AR representations augmenting the real appliance, which the user can directly interact with. For example, we show a volume slider above a sound system. The user can select the volume level by looking at it and then employ the pinch gesture to hold and drag the slider and alter the volume level.



(a) (b)

Figure 4.3: Left: The beginning of the pinching gesture. Right: The gesture while holding a control element, e.g. a slider.

Scenario 2 Our second scenario is also based on the direct manipulation of the AR controls by employing gestures. However, we allow for potentially more freedom in the way the gestural interaction takes place by utilizing a broader and more natural gesture set. We chose an electromyography (EMG) armband, the Thalmic Labs Myo (cf. Figure 4.4), for this purpose. The advantage of EMG over other forms of gesture tracking is that the arm and hand can be in any pose and do not have to be in the field of view of any camera. We chose the Myo because it recognizes gestures user-independently, i.e. without having to train the recognition per user. It is worn on the forearm, measuring the activation potential of the muscles controlling the hand and fingers, thereby being able to distinguish different hand and finger gestures. The Myo can recognize five predefined, user-independent gestures: Waving one's hand to the inside (*wave in*) or to the outside (*wave out*), making a fist (*fist*), spreading one's fingers (*spread*), and double-tapping thumb and middle finger (*double tap*). Furthermore, it contains motion sensors to measure acceleration and rotation around all axes. We use the *fist* gesture to activate the parameter control. The user can then set values by rotating his/her arm. We also use the waving gestures for discrete input. Our representation informs the user, which gesture he/she may use to perform which action. For example, Figure 4.5 shows the AR view instructing the user that he/she can either wave to the right to play the next song or make a fist to set the volume. Upon performing the fist gesture, a continuous value setting appears as shown in Figure 4.1(2), telling the user that a clockwise rotation of the arm increases the volume level and a counter-clockwise

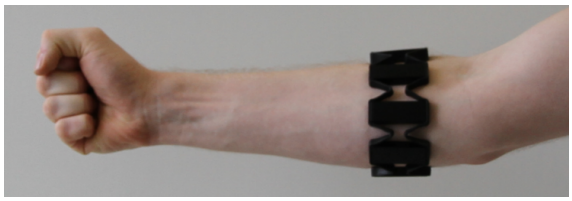
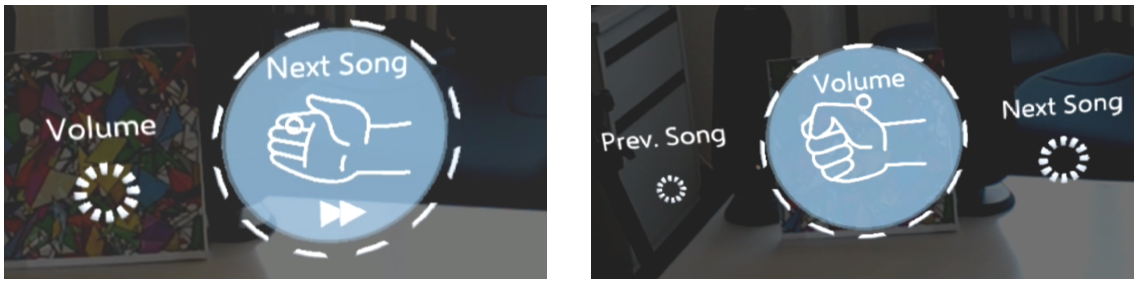


Figure 4.4: A user wearing the Myo and currently holding a control by making a fist and adjusting it by rotating the arm.

rotation decreases it.



(a) (b)

Figure 4.5: An AR representation augmenting a sound system, informing the user about possible gestures and corresponding actions.

Scenario 3 In the third scenario, we explore the idea of using objects in the user's environment to control devices. These potentially simple, unconnected objects can be moved or rotated by the user. The movement is registered by our system and thereby allows input by simply manipulating the object, now providing a tangible user interface. In comparison to the two other scenarios which allow the direct control of the devices, the object becomes a remote or proxy control which additionally allows us to evaluate the influence of this characteristic on the participants' perception of the system. For our study, we use a simple tea mug, to which we attached a visual marker to be tracked with the HoloLens' camera as depicted in Figure 4.6. The mug has a circular shape and naturally can be rotated to change the volume of a sound system, for example. Additionally, it can be moved in several directions which might be used for other triggers. As we do for the Myo gestures in Scenario 2, we inform the user about the possible interactions through AR visualizations, i.e. which movement of an object will cause which effect. Moreover, we show the current state of the appliance in AR overlayed on the mug. Figure 4.1(3) shows a user adjusting the volume of a loudspeaker by rotating the mug. Note that in order to track the movement of the mug, its marker has to be in the field of view of the HoloLens camera, which is a limitation as a user might rather want to look at the appliance he/she is currently controlling than at the mug. This is a general limitation, as either the object has to be tracked by an external camera, or contain motion sensors itself.

4.3.2 Implementation

The main component of our system is a Microsoft HoloLens, which tracks the three appliances and the mug used in Scenario 3 by detecting visual markers attached to them.

Figure 4.6: The tea mug with a visual marker for recognition used in the third scenario.

We chose the HoloLens because, at the time of the study, it was the only high-quality HMD which fulfilled our requirements of having a see-through display and robust and accurate inside-out tracking to place the interaction elements in space. For recognition and tracking, we use the Vuforia SDK [235]. All the visual AR representations are implemented in Unity. In our prototype, the primitives are fixed for the specific devices. However, they could be described in an abstract form in an XML document, downloaded from a web server on demand and parsed in order to automatically create the representation of the appliance as Mayer et al. have shown [125]. The HoloLens controls all the appliances by sending commands via a Wi-Fi network. For the purpose of exchanging state information and commands, we implemented a simple message protocol based on JSON. To simplify the control of the appliances, we use an Android smartphone (a Nexus 5X running Android 8.1), which receives the commands from the HoloLens and forwards them to the devices. The smartphone itself also acts as a sound system by being connected to speakers (the smartphone is not visible to the participants). The smart lamp, produced by LIFX [231], can be directly controlled by sending UDP packages over the Wi-Fi network. The smartphone is also used to receive the gestures from the Myo armband via BLE in Scenario 2. These are in turn forwarded to the HoloLens. As we only carry out mockup calls, the telephone merely has to be recognized, but not controlled here. A schematic diagram of all the devices and their information flows is given in Figure 4.7. A desirable property of the prototype is that it only utilizes the head-worn computer and smart things connected over a Wi-Fi network. Therefore it is mobile and not confined to a restricted space.

4.3.3 Evaluation

Study Design

We compare the three interaction methods in an appliance environment consisting of a smart lamp, a sound system, and a landline telephone in an office, as shown in Figure 4.8. The lamp can change its colour in the HSV colour space, i.e. the adjustable parameters

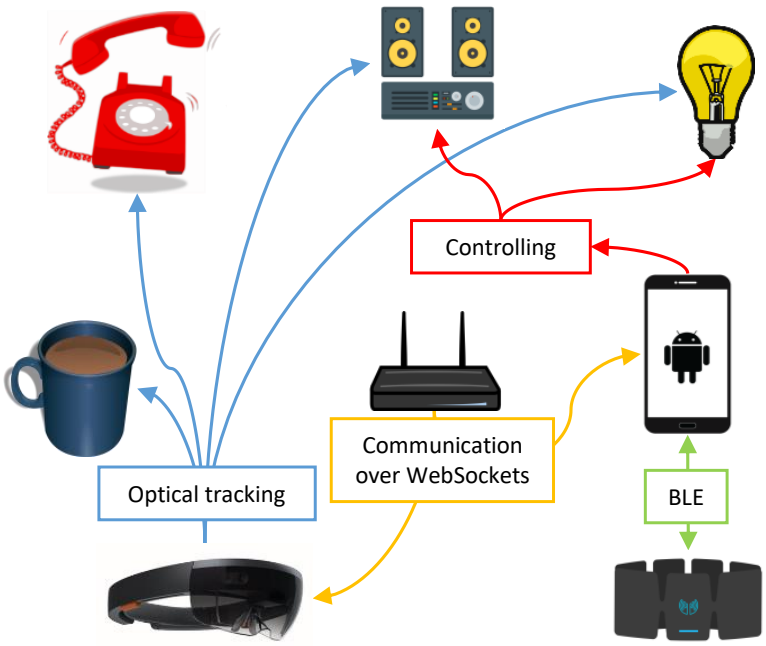


Figure 4.7: The HoloLens tracks the appliances and the mug (in Scenario 3). It communicates to the appliances and the Android smartphone via a Wi-Fi network. The Myo is connected to the smartphone via BLE.

are the hue colour angle, the saturation, and the brightness. It can be turned off by setting the brightness to zero. For the sound system, we preselected a set of three songs. The user can switch to the previous or the next song in the list and adjust the volume level. Similar to the lamp, the sound system can be turned off by setting the volume to zero. When selecting the telephone, the user can select a contact from a predefined list of five imaginary contacts and then start a call. Instead of actually calling someone, we provide visual feedback in AR by a telephone icon, that the call is in progress. During a call, the user has the option to hang up, which is also confirmed by visual feedback. This means the telephone is the only appliance which is actually not connected and actuated but only allows a mock-up call, which are shown in AR. Note that in our prototype implementation, the set-up is fixed to these appliances for this experiment; nevertheless, our concepts generalize to arbitrary appliances, which may even be located in different places.

At the start, each participant signed a consent form and was instructed about the purpose of the study by the investigator. Each participant carried out a fixed series of appliance interactions in each of the three scenarios. The interactions series were embedded into a story told by the investigator and included the following steps in the given order:

Figure 4.8: The set-up of connected smart devices we ue for our prototype implementation consisting of a lamp, a sound system, and a phone. The mug used in Scenario 3 is also shown. All have an individual marker attached for visual recognition.

1. Changing the light colour to green, setting the brightness to 30% and the saturation to 80%.
2. Turning on the sound system, setting the volume to 20, and choosing either the next or previous song.
3. Turning off the music.
4. Making a phone call (initiating the call).
5. Terminating the call.
6. Resuming the music player, switching to another track. Then the participants should change to another chair standing at a different position to the set-up (this is particularly interesting in Scenario 3 because they have to take the mug with them).
7. Increasing the volume of the music. Afterwards,
8. They sit back at the table and stop the music player.
9. Turning the brightness of the lamp to 50%.
10. Switching off the lamp.

Every participant carried out ten interactions per scenario, i.e. 30 in total. The order of the interaction within each scenario was the same; the order of the scenarios was random for each participant to avoid a learning bias. Before each scenario, the participants were briefly instructed on the new type of interaction. After finishing with each scenario, the participants filled in a questionnaire on the scenario containing the following questions, each to be answered on a five-point Likert scale from “disagree” to “agree” with a score ranging from -2 to 2 with the neutral score at 0:

Q1 The way how to control devices was easy to understand.
Q2 It was fast.

Q3 It was appealing.
Q4 It was not physically demanding.
Q5 It was not mentally demanding.
Q6 I felt comfortable.
Q7 I felt that I was in control of the devices.
Q8 I can imagine using this method in my daily life.

Additionally, the participants were given a User Experience Questionnaire (UEQ) [110] score sheet to obtain a standardized measure of the user experience. The questionnaire is available online [225]. The 26 items have a seven-point score ranging from -3 to 3. The UEQ analysis results in scores in the following six dimensions: attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty.

After having performed all three scenarios the participants fill in another questionnaire with pairwise comparisons (also allowing for a neutral decision) on which of the scenarios they preferred concerning the following questions (i.e. there are three pairwise comparisons per question):

P1 Which method did you generally prefer?
P2 Which method was faster to use?
P3 Which method was easier to understand?
P4 Which gave you the feeling of the most control?
P5 Which was more fine-grained?

Besides, we measured the task completion time from the moment the investigator instructed the participant to perform a task until it was fulfilled. One entire session took around 45 minutes.

Participant Population We recruited five participants for the pilot study (average age 24.0 with a standard deviation of 1.9, ranging from 22 to 26, two females) and 25 further participants (average age 29.3 with a standard deviation of 9.7, ranging from 21 to 52, six females) for the actual study. 36% of the participants had previous experience with VR (virtual reality) and 16% had previous experience with AR applications. The participants took part on a voluntary basis without compensation.

Results

We carry out a statistical analysis on the four measures: the questionnaire, the UEQ, the pairwise comparison questionnaire, and the task completion times. For all the four measures, the normality condition for applying an ANOVA test is violated (significant

Shapiro-Wilk's test and visible from QQ-plots). Hence we employ the Friedman test. If the Friedman test finds significant differences, we carry out post hoc tests employing a Bonferroni-corrected Wilcoxon signed-rank test to reveal where the differences lie. In general, we use a significance level of 0.05, in the cases using Bonferroni-correction 0.017, respectively. We only include the data of the 25 participants from the actual study and not the additional five from the pilot study. For the sake of shorter notations, we use the abbreviations S1, S2, and S3 for Scenario 1, 2, and 3, respectively.

Questionnaire Results The questionnaire, filled in after each scenario, asked the participants to answer the questions Q1 to Q8 in Subsection 4.3.1 on a five-point Likert scale ranging from -2 to 2. The average results for the 25 participants are displayed in Figure 4.9. The mean scores for all the questions are positive. For all questions except

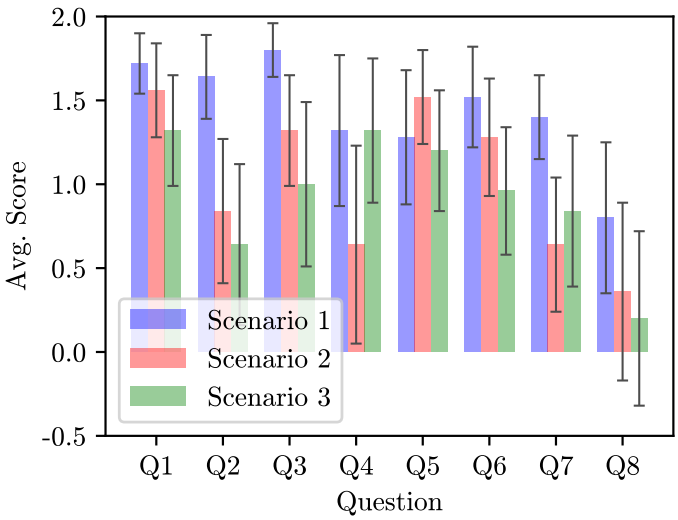


Figure 4.9: Average results of the Likert items in the questionnaire for all three scenarios with 95% confidence intervals.

Q5, S1 obtains the highest average scores. S2 mostly achieves higher scores than S3, however the only significant differences we find are between S1 and S2 (Q2 $p < 0.005$, Q3 $p < 0.015$, Q7 $p < 0.01$), and S1 and S3 (Q2 $p < 0.005$, Q3 $p < 0.005$, Q8 $p < 0.017$). The results indicate that on average, the participants have a more positive impression of S1 compared to the others. Especially for the subjective impression of speed (Q2), the scores are more than twice as high for S1 than for S2 and S3.

UEQ Results The results of the UEQ are six scales, calculated as averages from the items on the UEQ sheet, i.e. we can calculate a score for each scale for each participant and scenario. The average scores across all participants for these six are shown in Figure 4.10.

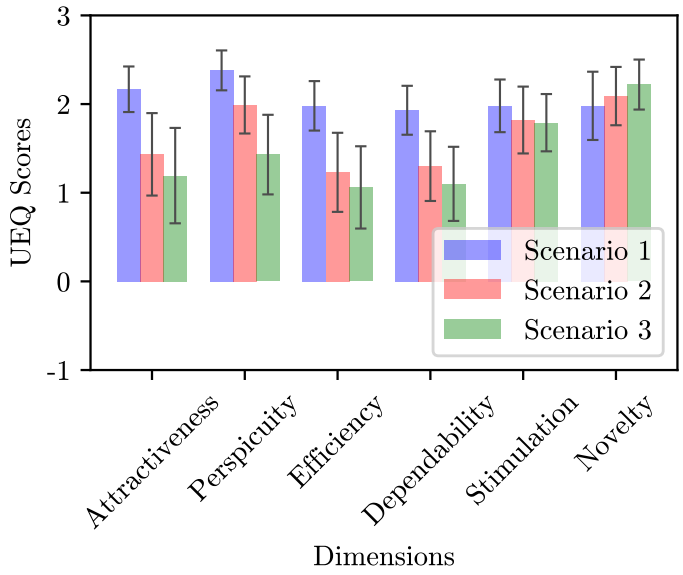


Figure 4.10: The average results of the UEQ, segmented in the six dimensions, for all three scenarios. The error bars show the 95% confidence intervals.

Again, all the results are positive. The results in the UEQ generally support the findings of the questionnaire mentioned above. All the average scores are positive, and again, apart from the scale Novelty, S1 has the highest average scores. For the first four scales we find significant differences between S1 and S2 (Attr. $p < 0.005$, Eff. $p < 0.005$, Dep. $p < 0.005$), and S1 and S3 (Attr. $p < 0.005$, Per. $p < 0.005$, Eff. $p < 0.005$, Dep. $p < 0.001$), only for Perspicuity the difference between S1 and S2 is insignificant ($p = 0.03$). Between S2 and S3, we find no significant differences. For Stimulation and Novelty, the means are similar (as can be seen from the plot), and we cannot find any significant differences.

Pairwise Comparisons The results for the pairwise comparisons are shown in Figure 4.11 as the sums over the participants' preferences per question item.

Across all question items and participants, there were no cyclic dependencies (such as S1 being preferred over S2, S2 over S3, and S3 over S1), i.e. it was possible to transform the pairwise answers into rankings for each question item. For these, we can carry out a Friedman test and post hoc tests as above. As for the other measures, S1 is mostly preferred over S2 and S3, which is also supported by the statistical tests (S1 - S2: P2, P3, P4 significant with $p < 0.001$; S1 - S3: P1 $p < 0.005$, P2 $p < 0.001$, P3 $p < 0.005$, P4 $p < 0.017$, P5 $p < 0.01$). Only for P5, S3 was chosen more often, i.e. the majority believe that S3 offers a more fine-grained control. Between S2 and S3, there were no statistically significant differences, apart from the aforementioned P5 (S2 - S3: $p < 0.001$).

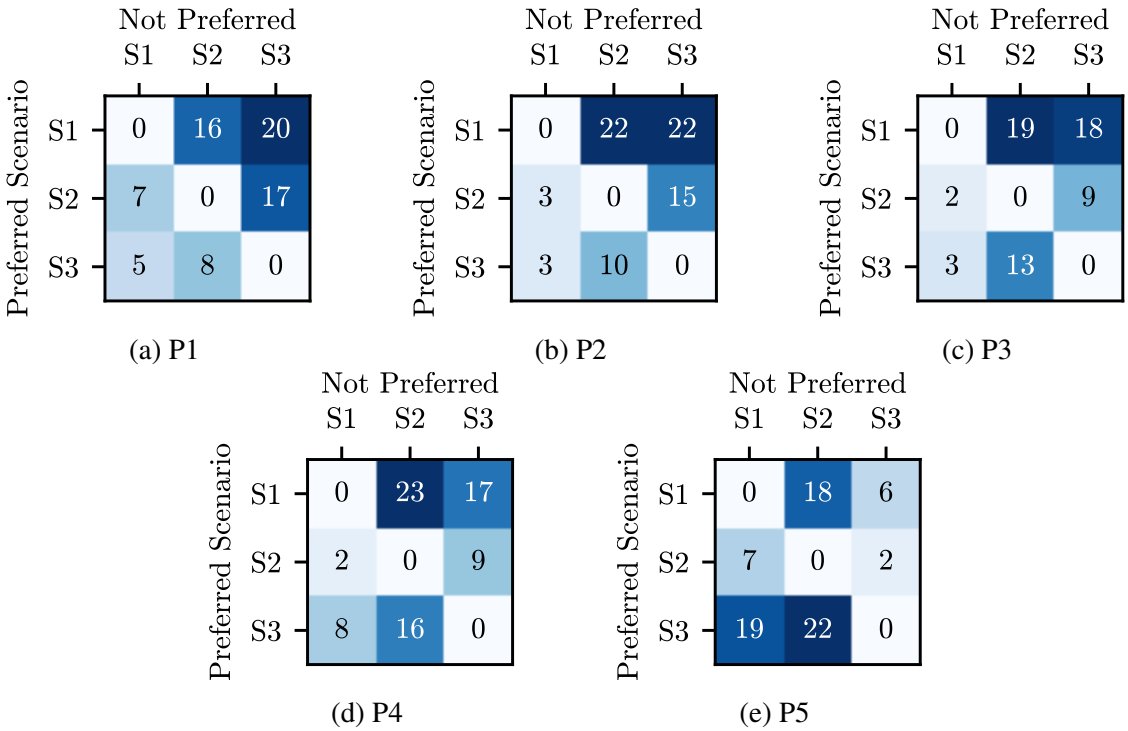


Figure 4.11: The results for each pairwise comparison item summed over all participants. The plots show how often which scenario was preferred over which other scenario.

Task Completion Times We calculate the mean task completion time per participant and scenario. Since task 6 and 8 included changing from one spot to the other, which constitutes a rather undefined period of time difficult to compare, we omit those tasks. The averages of all participants are given in Table 4.1. The average overall task completion time for S2 is nearly twice as high as for the fastest, S1. For S3, it is 50% higher than for S1. The differences in the overall mean completion times between all scenarios are significant (S1 - S2: $p < 0.001$, S1 - S3: $p < 0.001$, S2 - S3: $p < 0.01$). Interestingly, the order S1 $<$ S3 $<$ S2 holds in each task.

Table 4.1: Mean completion times in seconds for each measured task over all participants for each scenario.

	T1	T2	T3	T4	T5	T7	T9	T10	all
S1	31.8	9.0	21.2	3.5	3.6	1.2	8.4	1.6	10.0
S2	51.5	17.0	42.4	4.8	5.1	5.6	20.2	6.1	19.1
S3	41.1	12.3	32.6	4.0	4.9	5.4	17.6	4.2	15.3

4.3.4 Discussion

As the positive scores in the questionnaire (e.g. Q3) and Attractiveness in the UEQ show, our participants liked our system in general and were excited to use it (high scores for Stimulation), most were even amazed at this new way of interaction. We received many comments, such as "It was a great experience", "So cool, this is amazing" and "I am mind-blown". Besides the general appeal, all participants found the system to be innovative.

General advantages of our approach which apply to all three scenarios are that it makes appliances easily accessible, as the user only has to look at them in order to control them and shares the benefit of remote controls by enabling an interaction from a distance. Furthermore, our interaction approach was easy to learn in all scenarios, as the scores for Q1 and Perspicuity indicate. We believe this to be the result of the possibility to show interaction instructions in a first-person view, e.g. which way the user should rotate his/her arm to increase a level. This makes the control very intuitive, as none of the interactions has to be remembered. Moreover, as the AR representation is locally collocated with the appliance (except for S3), there is no need to switch one's gaze between a remote control and the controlled device as it is the case for standard remote controls. This also makes it easy to understand which appliance is currently affected by changes. Several participants stated that "it is easy to get used to".

When comparing the three scenarios, we find a strong preference for S1, indicated by the questionnaire, the UEQ, and especially by the strong overall votes for S1 in the pairwise comparisons. Appliance control was also significantly faster on average than in the other scenarios. This objective measure aligns with the subjective impression of higher speed, as shown in Q2 and Efficiency. Many participants expressed their liking of S1 because of its simplicity compared to the others, which also made it easier to understand: "Easy to understand", "Intuitive", "worked like a charm", "more intuitive and faster", "good that you only have to learn a single gesture", "I don't have to think too much", "really intuitive, although I'm not a geek". One important reason we believe for this scenario being so popular is that it provided the most direct interaction because one is virtually moving sliders and clicking buttons on the appliance itself. One participant described the reason for his preference for S1 accordingly as "the feeling of the direct interaction". The results for Q7 and P4 also show a significantly higher score for the feeling of being in control of the devices, supporting our assumption.

The only major shortcoming of S1 was the difficulty to adjust the values in a fine-grained manner. Participants stated it was easy to set an appliance parameter to the minimum or maximum, but rather difficult to set it to an exact intermediate value. This is also exhibited by the preference of S3 over S1 in the pairwise comparison item P5 on which scenario

offered the most fine-grained control.

S2, using the Myo for gesture recognition, received a higher score than S3 for many of the questionnaire items. However, only the difference in the scores for P5 was significant, but then in favour of S3. What participants valued, similarly to S1, was the direct interaction with the appliances. Moreover, some preferred the use of natural gestures, such as waving to the left or right, over the clicking on buttons and dragging sliders in S1: "The coarse-grained gestures in S2 are better than the fine-grained gesture in S1". Furthermore, a participant mentioned this form of gesture input is better because one does not have to raise the hand to be in the field of the camera, but it can be anywhere. Another participant even said that he thinks this would be faster than S1 because one does not have to raise the arm. One interesting aspect is that the subjective impression of speed does not match the objective task completion times. There are slightly higher scores for the question items on the speed for S2 than for S3 (but not significant), although the task completion times show that S2 is significantly slower than S3.

In comparison to the two other scenarios, there was a great conceptual difference in S3. Instead of direct gestural control, S3 offered a tangible user interface through the movement of a mug. Participants found this scenario very innovative and interesting, but at the same time not as suitable for appliance control: "more innovative than S1, but also more cumbersome" or "very cool, but liked much less than S1". The participants strongly prefer the form of direct control that S1 and S2 offer, as mentioned above. Some also expressed their dislike of the necessity to frequently change the viewpoint back and forth from the appliances to the mug. In our implementation, the mug is only tracked visually by the HoloLens, which requires it to be in the field of view of the HoloLens' cameras. This means the user has to be looking at the controlling objects to be able to perform actions and cannot look at the controlled appliance at the same time, which was very unnatural for many participants. Several participants said that looking down is uncomfortable when wearing the relatively heavy HoloLens. This limitation could be solved by visually tracking the object with an external camera. However, this defies the goal of a mobile application.

On the other hand, one strong advantage S3 has over the other scenarios, is that it offers haptic feedback, as one participant expressed, for example: "the haptic feedback is great". This goes hand in hand with the fact that S3 was on average preferred for setting the parameter values in a fine-grained manner, as shown by the results for P5. The participants showed strong interest in this scenario and understood our intention to potentially use any object as it "would be great if this worked with all objects", and one would not have to take it with oneself.

In summary, we found a strong preference for the simplest way of control, S1, using

only a single gesture together with comprehensible visual representations. According to our participants, this offered the most direct form of control. A similar scheme of interaction was given in S2, however, with more gestures. This appeared to be significantly slower and more physically demanding. S3 affords a novel interaction paradigm, which provides tangible feedback through the physical object. The participants indeed found this interesting. However, our evaluation shows that the more direct form of control is preferred, which does not use an interaction proxy. Nevertheless, the haptic experience of the interaction allows for a fine-grained adjustment.

A conclusion we can draw when contemplating all scenarios is that an AR representation offers the benefit of allowing for direct control of appliances without requiring any proxy controller device in between. We believe this is also helpful advice for the design of potential future human interfaces of appliances.

4.3.5 Limitations

The general positive attitude towards our approach is despite the HoloLens being relatively bulky and heavy, and furthermore having a little field of view. These constraints currently are a limitation to the practical use of our approach and are reflected in the relatively low scores for the question, whether participants could imagine using our system in their daily life (Q8). Nevertheless, we believe there to be more practical AR devices in the future with a broader spread among the general public. We envision an AR system, which is built into a person's glasses, ubiquitously available, making our approach also practical for daily life.

A further limitation of our work is that the prototype system, including the interfaces displayed, was fixed to three devices and to a single office. The concept itself is applicable to many more devices and a much larger space. As mentioned before, we designed the interfaces in a way that they could be used in a generalized form, which could be automatically generated from abstract device descriptions, as shown in [125]. Especially the extension to a larger space with more rooms would be an interesting continuation because then also proximity-based interactions as in [86, 111] could be explored. In this sense, another limitation is that often the devices were within reaching distance for the user. This raises the question of whether the widget augmentation and the use of gestures is necessary at all. The main reason for this is that the recognition of the markers does not work well at a large distance without significantly increasing the size of the marker. Hence it was limited in our study. This limitation could be mitigated by implementing object recognition based on the appearance of the devices (which would eliminate the necessity of having to use markers overall), or using additional technology

such as infrared emitters and receivers to transmit IDs between devices and the HoloLens (augmented with an IR sensor) which works across a range of several meters [19, 37, 193]. However, the implementation of these more complex approaches is out of the scope of this work. Nevertheless, in some situations, the appliances were clearly out of reach for the participants, especially after moving to another chair. Furthermore, the participants never questioned the interaction design in spite of the proximity of the appliances. In any case, we believe that the widget augmentation and the interactions we present here still have benefits compared to interfaces on the appliances themselves, in particular, because the appliances are not required to have an interface of their own.

Another limiting factor could be the fact that 15 out of 25 participants had no previous experience with VR or AR applications which might bias these participants in their general assessment of the interaction system by being amazed by the AR experience in general. Nonetheless, also among the participants with previous VR and AR experience, most were positively surprised by the way our AR approach supported device interaction and also explicitly stated this in their comments. In fact, there were little differences in the attractiveness scores between the two groups for example (only S1 having a significantly higher score for the inexperienced group). In the comparison of the scenarios, there should be no bias, as none of the participants knew the system beforehand.

4.4 Device Network Configuration through Wearable Augmented Reality

Hitherto, the concept we presented only allows controlling a single appliance at a time. A significant part of the time spent on dealing with appliances may, however, be due to the configuration of multiple devices. We can expect this proportion of effort to increase over time as more appliances can be interconnected to others. For example, a smart switch could be connected to a smart light. This flexibility has the benefit that the configuration can be changed, in contrast to conventional switches with hardwired connections. We could even see the Wi-Fi connection of a router to a laptop as such a connection which can be reconfigured, e.g. the network owner might want to interrupt the connection. Building upon our results in the first part, we present *ConnectAR*, a prototype system fulfilling our goal of allowing such a reconfiguration of compatible smart devices in an egocentric AR view. Here, we focus on connections between devices and controllers, i.e. components which can be manipulated by the user and then relay this manipulation to the connected device, and not arbitrary connections between devices. ConnectAR also avoids a limitation of our work mentioned above: at least at the current time, users may not be willing to

wear an AR headset all the time in order to be able to control appliances. Now, this is not necessary anymore, as it is only used for configuration.

Our idea is to enable logical control connections between smart devices and controllers and to allow the configuration of smart devices and controllers in the visual AR space. For example, we want to be able to connect a smart light bulb to a switch by simple manipulation in AR. The user can see all existing connections in his/her egocentric view, but also create new or modify connections by employing the simple pinching gesture, which we found to be the most favourable option in the first part. Naturally, we do not only require the devices to exhibit "smartness", but also the controllers by being able to connect to a Wi-Fi network and providing an interface to transmit their value settings, i.e. the devices would have to abide by a common standard to fulfil our concept. We envision this to be possible in the future. For our prototype, we use devices which we can freely modify to enable a common interface. The devices and controllers are visually recognized with the HoloLens' camera by their individual markers as for our first prototype. Marker-based recognition suits our prototyping purpose very well. As already mentioned above, in the future, we could integrate visual object recognition or a device selection based on infrared light as proposed in [19, 37, 193].

While wearing the HoloLens, the creation of a new connection can be initiated by looking at a component (either a device or controller), more precisely at its marker, which is then recognized. The gaze is approximated by the head pose provided by the HoloLens. The user may then select the component for a potential connection by performing the finger pinching gesture, which is recognized by the HoloLens. A selection is indicated by a scanning symbol (cf. Figure 4.12) while the system is waiting for the selection of a second component. This counterpart component, a controller for a device or vice versa, is selected the same way, and we then establish a logical control link between the two. This connection is visualized by a line between the spatially mapped controller and device in the visual AR space, as shown in Figure 4.2, for example. Instead of drawing a straight line between them, we interpolate a line along the positions of the HoloLens during the connection process since there might be obstacles in between the components in the way of a direct line, or the components might even be in different rooms as the user may have moved from one to the other. Each visual connection has a deletion option which can be executed by tapping on the virtual red sphere located in the middle of each connection line.

The question that remains is how to check the validity of potential connections and how to convey the control information without hardcoding it for each pair of device and controller, which would result in a potentially large number of mappings. Similarly to Mayer et al. [125], we create machine-readable and downloadable interaction abstractions for each device and controller, which can be retrieved on demand and interpreted. An

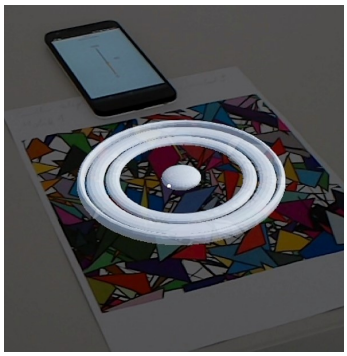


Figure 4.12: When a controller marker is recognized, a scanning symbol is displayed, notifying the user that the system expects a device to be selected.

abstraction defines which capabilities and requirements a device or controller has. For example, the abstraction for a smart light bulb which only has a brightness setting could specify that it can handle continuous values, the abstraction for a simple switch only on/off. Hence, *ConnectAR* has to interpret the two abstractions and match them in the best possible way and thereby defines the relationship between device and controller. In the example above, this could mean that the on/off-function of the switch is matched to the maximum brightness and turning the light bulb off, respectively. If no match can be found, the potential connection is invalid and not allowed. This is indicated to the user by a cross symbol (cf. Figure 4.13).

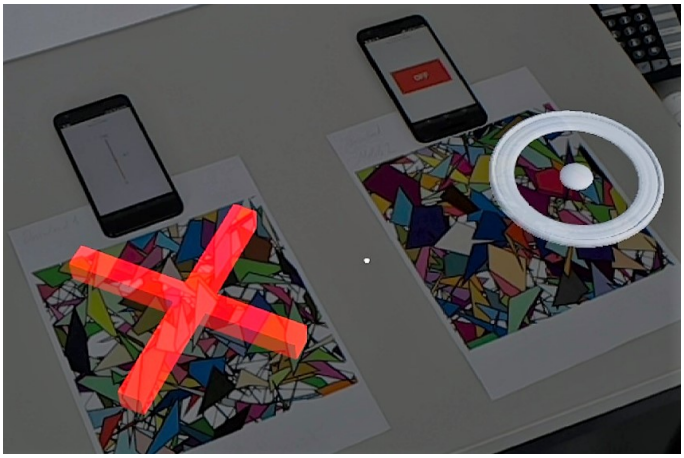


Figure 4.13: A cross symbol shown when no valid connection is possible, e.g. when trying to connect two controllers.

We use the following simple set of abstractions for describing the component capabilities and requirements:

1. `k states`, representing `k` possible positions on a one-dimensional scale, e.g. a simple switch could provide two positions which could be used for an on/off functionality.

2. `float value from lower_limit to upper_limit`, e.g. as a slider could provide.

Although we use only two abstractions, we can already provide a large space of functionality. Each function of a device and a controller can now be described by the abstractions. For example, the smart light bulb we use in our prototype can change its appearance in the HSV colour space, i.e. there are three functions: setting the hue colour angle, the saturation, and the brightness value. An abstraction specification could look as follows:

```
hue float value from 0 to 360
saturation float value from 0 to 1
brightness float value from 0 to 1
```

For a simple controller, a switch with only two states, the specification could look like the following:

```
switch 2 states
```

The switch cannot fulfil the full specification of the lamp. However, we try to match them partially. In the given case, for example, the on/off-switch could set the brightness value to 0 or 1, respectively. We divide the one-dimensional continuous value space into $(k-1)$ equal intervals and assign the ith out of the k positions the ith interval border. After creating the virtual line in AR, we let the user choose which characteristic (e.g. hue, saturation, or brightness) the switch should be matched to by holding the pinching gesture and dragging the virtual slider to the intended position as shown in Figure 4.14.

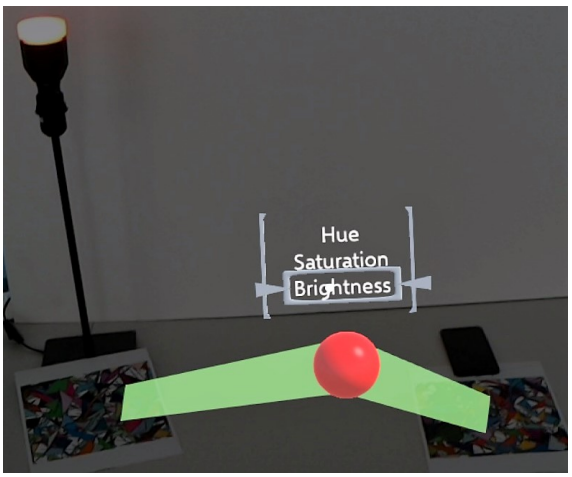


Figure 4.14: The connection settings defining which characteristic of the device is controlled.

For valid connections, we allow any form of combinations of devices, i.e. many-to-one combinations (many controllers, one device) are possible as well as one-to-many combinations (one controller, many controlled devices) or any other combination (several controllers controlling overlapping sets of multiple devices). For example, the same light could be connected to several switches, responding to all of them or a single switch could be connected to multiple lights, controlling all of them at the same time.

The message exchange between the controller and the device is similar to the abstractions for creating the connections. The messages contain the abstractions concatenated with the current state. For example, upon being adjusted to level 6, a slider controller might send the following message:

```
slider float value from 0 to 10:  6
```

For the ease of implementation in our prototype, we mainly use controllers implemented using Android smartphones (Nexus 5X with Android 8.1) running an app which displays simple control elements such as an on/off-button or sliders for setting continuous values as shown in Figure 4.15. However, if available, we could include any kind of controller which implements our protocol, e.g. a knob with a Wi-Fi connection.

Nevertheless, the availability of the spatial tracking capability of the HoloLens makes it possible to also incorporate simple, unconnected objects as controllers as in Interaction scenario 3. The tracked movements of the object can then be used for control. As smart devices, we incorporate the light bulb which may be controlled via Wi-Fi and can change its colour in the HSV space, and a sound system for which the volume can be adjusted

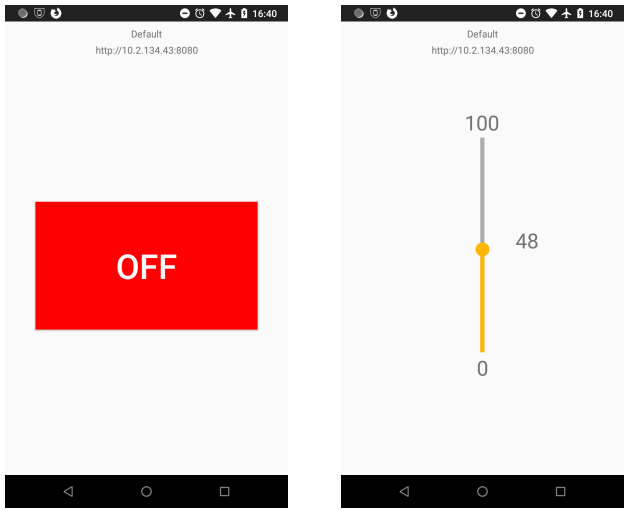


Figure 4.15: Screenshots of two control elements in our prototype controllers, an on/off-button and a continuous slider.

as for our prototype. A desirable property of the prototype is that it only utilizes the head-worn computer and smart things connected over a Wi-Fi network, therefore it is mobile and not confined to a restricted space.

The HoloLens controls all the appliances by sending commands via a Wi-Fi network which is established by a router. For the purpose of exchanging the validity messages, state information, and commands, we again implemented a simple message protocol based on JSON. To enable the control of the appliances and the message exchange, we use an additional Nexus 5X handling the light bulb via UDP and being connected to loudspeakers and thereby providing a mockup of a sound system (the smartphone is not visible). A schematic diagram of all the devices and their information flows is given in Figure 4.16, which is similar to the schematic for our first prototype extended by the logical control connection. Currently, all logical connections and settings are stored in the Hololens. Whenever the user deletes a connection, it is simply removed from the connection list and thereby the interaction between the corresponding component automatically ends. The connection information could also be stored on the network router in the future, to make it possible to manage connections from several devices or share these with other users. We are aware that our approach would require explicit rights management in the background to avoid unauthorized manipulations of the overall system, and we leave this to future work. However, here we also see the potential for our concept to make it easier for users to understand access rights, as access levels can easily be visualized and inform users when they infringe upon the stronger authorization of others.

4.5 Conclusion

We presented two concepts for the interaction with smart devices through their representation in a head-mounted AR display and implemented them in two prototypes. In the first part, we presented an interaction paradigm in which we externalize the user interface of appliances into visual representations in AR, using a head-mounted display. These representations provide an overview of the properties and capabilities of every appliance in the user's first-person view and also allow appliance control. We implemented three different ways of interaction: (1) using simple gestures to adjust virtual sliders and click virtual buttons, (2) employing more gestures recognized by an EMG armband, and (3) extending the representation to an object, in our case a tea mug, which can be utilized for control by moving it and thereby provides a tangible interface. We conducted a user study in an environment consisting of three appliances, and our evaluation shows that the participants prefer the direct control and simplicity featured by the first scenario, which is also significantly faster than the other two options. Although the use of the object may

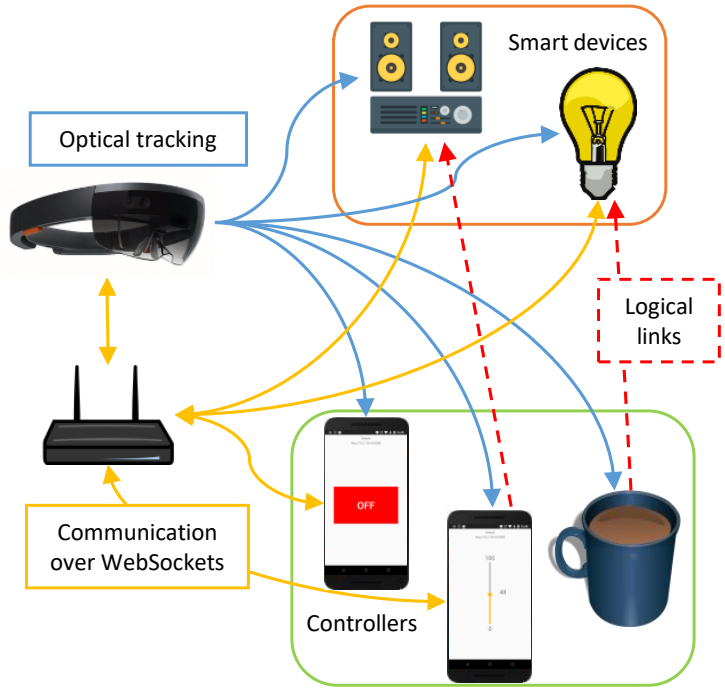


Figure 4.16: The HoloLens tracks the smart devices and the controllers. It communicates to the devices and the controllers via a Wi-Fi network, which is established by a router. The logical control link between a controller and a device is established once the user selects these and their potential connection is valid.

seem the most intriguing interaction possibility, it introduces the object as a proxy control. This lets the user interact with the object, rather than with the appliance, which was less preferred by the participants.

Nevertheless, this first concept cannot capture a many-to-many-relationship of multiple devices that work together. We built upon the results in the first part and designed a second approach for configuration of control connections between smart devices and controllers in AR. Connections can be created and modified directly in the AR space by employing simple gestures. We propose an interface abstraction to represent devices and controllers and thereby are able to verify the validity of a potential device connection. This abstraction also allows simple, unconnected objects to be incorporated as a controller by coupling the virtual representation to their physical movement. We envision a method such as our prototype to be built in the operating system of head-mounted AR displays, which subtly shows the configuration options whenever a known configurable device is recognized. One goal of our concept is to break up the fixed coupling of controllers and devices, e.g. a specific remote control for an appliance or a specific switch for a single light, and set free their functionality for any valid combination among them. By using a head-mounted AR display and only wireless communication over a Wi-Fi network, the system is mobile, and

the user is free to move without limits.

One interesting area of future research would be to extend the configuration possibilities from a pure controller-actuator relationship to data producer-consumer relationships with logical rules in between, such as between a light sensor and the blinds or a lamp. The lamp could react to the values it receives from the sensor, turning on in darkness and off at daylight, for example. We could visualize such data streams in AR and also allow their configuration, thereby providing a simple "programming" interface for devices in AR. Another way to enhance our approach could be the use of spatial sound attached to the devices or the use of speech commands. Furthermore, we could take the proximity of already mapped surrounding devices and controllers into account by, for example, showing the nearest compatible devices and controllers or displaying connections based on proximity.

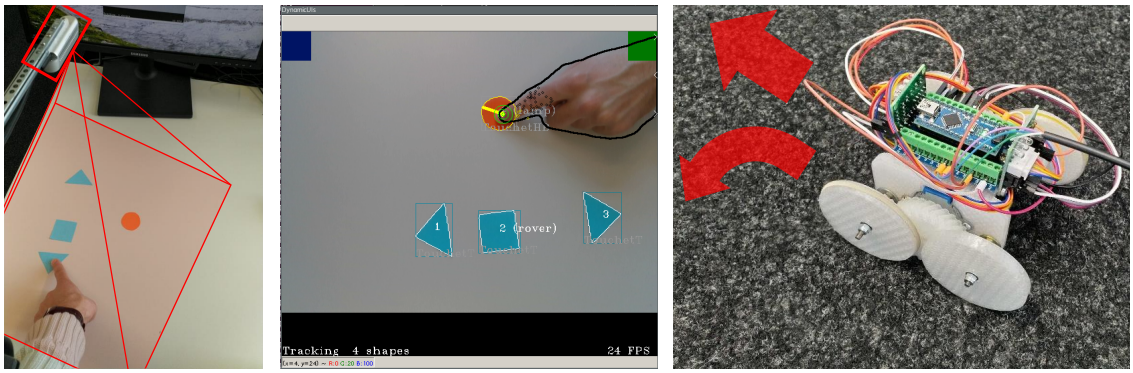
Chapter 5

Creating Personalized Tangible User Interfaces from Simple Materials

5.1 Introduction

In the previous chapters, we introduced different user interfaces, either purely gestural ones or through representation in augmented reality. Similar to most other interfaces, these do not adapt to the specific user preferences or the task at hand. In this chapter, we present a method that allows the quick and inexpensive creation of personalized interfaces from plain paper. Users can cut out shapes and assign control functions to these paper snippets via a simple configuration interface. After configuration, control takes place entirely through the manipulation of the paper shapes, providing the experience of a tailored tangible user interface.

Tangible user interfaces, such as buttons and knobs, enjoy widespread use for controlling appliances: they naturally arose as controls of purely mechanical devices before the emergence of digital products, and virtual implementations of the interaction patterns they express are ubiquitous in graphical user interfaces (GUIs) for software as well. In fact, with the spread of displays in the past decades, more and more interactions have been transferred from tangible mechanical controls or remote controls to GUIs. These have the advantage of being able to display changing content and therefore are highly flexible and widely applicable. However, GUIs often do not induce any haptic sensation, which humans innately prefer over passive interfaces [205]. Moreover, GUIs usually do not provide common physical interactions humans are used to, such as easily moving, adding, or removing elements. This has given rise to the creation and proliferation of tangible user interfaces (TUIs) in the past two decades with the aim of making interactions more natural



(a) (b) (c)

Figure 5.1: We present a concept and a prototype of a system for creating personalized user interfaces from paper. Users can cut out preferred shapes from paper and add interaction functionalities to them. The paper shapes and the user's fingers are tracked by an RGBD camera mounted above the interaction surface (Figure 5.1a). Our processing pipeline recognizes and measures manipulations of the shapes (Figure 5.1b), such as movements and touches, which can be mapped to certain functions according to the assigned interactions, e.g. for controlling a toy rover as illustrated in Figure 5.1c.

and binding virtual elements and functions to real objects. TUIs, which can take almost any physical form, are also conducive to the fundamental goal of ubiquitous computing: making computers (and their controllers) disappear physically and mentally into users' environments [180].

However, TUIs usually cannot adapt to a specific user nor to the task the user intends to carry out, simply because they are *hardware* devices and are thus constrained to – and by – their physical form. Consequently, many tangible interfaces that we use in our daily lives (e.g., traditional TV remote controls) are often built to provide interactors for all functions of the controlled device, even if these functions are used only rarely – or never. This can be overcome with interfaces that are able to change their shape during interaction [68, 98]. However, this currently requires complex and expensive electro-mechanical systems.

An intriguing solution for adaptive user interfaces is to let users themselves create and assemble their personalized user interfaces from inexpensive everyday materials (e.g., paper, cardboard, or playdough), and enabling them to link these TUIs to device functions. This would allow users to create exactly the interfaces they require, at their desired level of complexity.

Such an approach enables various applications. First, it benefits the process of user interface prototyping, where paper snippets are commonly used, by making it possible to add actual functionality to the paper shapes and thus being able to rapidly test designed interfaces – this enables us to bridge the "ideation" and "implementation" steps in the

design process, which are typically separate [39]. Furthermore, the interfaces can easily be reconfigured, thereby enabling quicker iterations in the design process and an earlier realization of potentially erroneous approaches as the UI does not have to be implemented first, which entails high development cost.

Second, an approach like this could be used in various real applications, for example, in the domain of building control, where it would permit the creation of *passive* control interfaces for lights, blinds, AC, and other devices that are attached to surfaces and walls without requiring infrastructure such as power and control lines. This would provide an installation that can be easily reconfigured, which becomes necessary as building automation companies are searching for ways to flexibly rearrange the spaces we live and work in, in response to user preferences.

Third, the development towards the increasingly automated operation of industrial equipment entails less frequent direct interactions of workers with machines in industrial shop floors. In the wake of this process, adaptive, *tactical*, user interfaces would enable us to avoid cluttering workshops with UIs that are operated infrequently (and might not even be suitable for a particular user). Rather, they would enable workers to bring their own interfaces with them, quickly assemble and configure them, and use them to interact with machines throughout the production space.

We present a system that allows users to create their own, personalized, user interfaces by cutting out arbitrary paper snippets and assembling them on a suitable surface (e.g., a table). We track both the shapes and the user's fingers with an RGBD (colour and depth) camera without the need to employ any markers, enabling a natural control of devices and permitting a wide range of interaction abstractions. In addition, our system allows for dynamically adding and removing paper snippets from the interface.

We determined the types of interactions that such a system should enable through an elicitation study with 20 participants, and deduced 25 abstract interactions. Out of these, we implemented the 13 most common ones in our prototype – these interaction abstractions cover more than 92% of all observed interactions from the elicitation study. When creating an interface using our system, users cut out paper shapes and assign interaction abstractions to them. Being tracked by the RGBD camera, these shapes respond to corresponding user interactions by emitting events that carry the relevant interaction information (e.g., the *distance* a shape was moved; the *angle* it was rotated; the *duration* it was tapped, etc.). These events are subsequently passed to any (networked) consumer where they trigger application functionality – this can be configured using a common dataflow programming language. We evaluated our complete prototype in another user study, which showed that participants were able to use our system to successfully create fully functioning interfaces for several different applications, such as controlling a sound system.

5.2 Related Work

Already in 1995, Fitzmaurice et al. [67] presented *Bricks*, an approach that connects physical bricks, laid out on a display surface, to virtual graphical counterparts depicted on the display. The virtual shapes can be manipulated, e.g. moved or rotated by manipulating the corresponding bricks. Shortly after, Ishii et al. [87] presented their vision of *Tangible Bits*, aiming at a coupling of digital information to physical objects and thereby making this information tangible. This coupling should also allow the manipulation of the digital state by the manipulation of the objects. Our approach fits this concept well by coupling components made from simple and inexpensive material, in our case paper shapes created by the users themselves, to digital control components.

Dedicated Tangible Interfaces Complex tangible interfaces that feature high representation capabilities are enabled by the interface changing its shape. For example, Follmer et al. [68] created an interface consisting of an actuated matrix of poles which can take arbitrary three-dimensional shapes. It can thereby provide tangible controls to the user or even actuate objects lying on it by dynamic shape changes. This concept is combined with an augmented reality (AR) application in [112] to add a tangible representation of the AR content through the actuated pin array. Shape-changing tangible interfaces can also be used to combine different elements, such as sliders or knobs in a single component [98]. Studies on shape-changing interfaces can be found in [49, 99]. These interfaces provide an interesting approach in the area of dynamic and adaptable interfaces. However, they also entail relatively complex hardware and high cost. We intend to follow the opposite direction. Users should be enabled to create their own personalized interfaces from simple, inexpensive material, and should be able to change and recreate interfaces themselves easily.

Printable Circuits One possibility for using paper for interaction elements is to print electronic circuits onto it. A number of projects have recently explored combining paper with electronics. For example, *Instant Inkjet Circuits* [91] and *Printem* [136] are simply created by a modified printer. The projects *Sketching in circuits* [141], *PaperPulse* [143], *CodeCollage* [142], and *LightItUp* [81] specifically explore how users combine various interactive paper electronics elements. The recent *Pulp nonfiction* [202] extends regular paper with capacitive touch and pen sensing. Some paper electronics are even available in commercial products such as Chibitronics [217] or DynamicLand [244]. The fundamental difference to our approach is that the paper, on which the circuits were printed or glued or stuck to, has to stay in a single piece, i.e. the user cannot create independent interaction

elements which can be moved or reassembled. As such, the paper circuits act more as printouts of graphical user interfaces than tangible interfaces.

Using Everyday Objects as Interfaces In addition to the creation of dedicated tangible user interfaces as mentioned above, arbitrary objects in the user's environment can also be used as tangible interfaces. Pohl et al. [138] investigated the space of everyday objects that could be useful as tangible controllers and found that there is both a diverse set of objects available and potential use for them. This concept is employed in several works in order to assign different functions to different objects, e.g. in painting applications [69, 159]. Often, a workspace is instrumented with overhead cameras to track simple objects and their movements and map them to control functions, for example, *iCon* [43] repurposes physical objects to control other objects by augmenting them with a trackable paper label. Corsten et al. [51] do the same without markers, using an over-head depth camera for pose estimation. Funk et al. [71] present an augmented workspace in which everyday objects can be combined into personalized control widgets. The idea of using objects as tangible interfaces is also used in mobile settings by employing tablets [11, 82, 83] or head-mounted AR displays as we showed in the previous chapter. In contrast to visual sensing, *Project Zanzibar* [170] contributes a portable rubber mat with embedded NFC readers that recognize various objects placed on top. This mat can be used to create reconfigurable tangible controls similar to our system.

Similar to our work, using simple objects in the user's environment also has the goal of using readily available components as TUIs at no (or hardly any) extra cost. Nevertheless, instead of using arbitrary objects, we intend to let users design their own interfaces – according to their personal preferences and abilities. Furthermore, our interfaces do not interfere with the normal use of objects (e.g., drinking from a mug that is also used as a "dial" TUI).

Reconfigurable Interfaces from Simple Materials Besides using existing physical objects as interfaces, there are other approaches which allow personalized interfaces and reconfiguration as in ours. Kelly et al. [93] recently presented *ARcadia*, which is intended for rapid prototyping. Control elements such as buttons, wheels, or sliders can be cut out from paper or cardboard, arranged and then assigned a certain function in a browser application. Fiducial markers are attached to the shapes to recognize and track them using a webcam, e.g. a laptop camera. The laptop can be used to configure the function assignments at the same time, as *ARcadia* requires an additional input device for this task. Using markers has the advantage of being able to track the shapes very accurately and giving the shapes a unique identity. However, the user has to provide the markers for every shape in the interface construction process, e.g. by printing them. In contrast to *ARcadia*,

our intention was to make the TUI creation process as simple as possible. Hence, we avoid the use of markers and directly track the shapes and the user's fingers. We furthermore enable a broader range of interactions, as we (1) track the fingers separately from the markers, thereby allowing users to simply tap on shapes – or swipe over them – instead of having to cover an entire marker and (2) study which interaction behaviour is employed by participants in an elicitation study resulting in a wider set of possible interactions which goes beyond simple movements and rotations.

The *VoodooSketch* [35] system allows users to draw user interface elements on light-reflective paper that are recognized by an over-head camera like in our setting. Our system provides similar flexibility and ease of use. The main difference is in creating the widgets via drawing or cutting and the selection of supported materials.

Olberling et al. [134] rely on printable circuits as mentioned above but furthermore allow cutting interaction elements from the circuit-augmented paper. They presented a special wiring topology for multi-touch sensors that make them robust to cutting, so users can define and cut personalized UI elements. An advantage of our approach is that no special material is required. The sensing part of our system, an RGBD camera, can be reused and any kind of paper can be used for cutting out shapes. Moreover, with the camera tracking, our approach allows the position and the movement of the widgets to be taken into account when defining controls.

Object Tracking An important component of these approaches is the recognition and tracking of the elements, which may be printed circuits or real-world object, as mentioned above. There are various technologies for tracking user-defined elements on a surface, all having their advantages and disadvantages. With cameras, for example, *ShadowTracking* [62] recognizes silhouette shadows from above a well-lit touch screen, *VoodooSketch* [35] and *RetroDepth* [97] apply special retro-reflective surfaces. *DIRECT* [188] combines RGBD and IR input. In our approach, we do not only track the position of the finger, but also its orientation. Instead of vision, also other modalities such as radar sensing [191] have been used. Voelker et al. [171] presented Passive Untouched Capacitive Widgets that can be tracked on top of unmodified capacitive touch screens. *SLAPWidgets* [183] provide tangible transparent silicon elements that can be reconfigured on the fly as various UI elements. The actual functions are projected onto the blank elements with an overhead projector. The elements are recognized from the bottom via IR-reflective identifiers. In comparison with our approach, the elements cannot be defined by the user in a simple manner.

5.3 Elicitation Study

Technically, our approach is based on visually tracking user-cut paper shapes. The manipulations users perform (i.e. movements, rotations, touches, etc.) are then mapped to specific value changes or discrete actions. For example, a user might create a paper circle intending it to be a knob he/she can use to control the volume of a sound system. The user should also be able to assemble basic shapes to groups, which then represent an interface element, e.g. one could use two rectangles and a circle to create a trackbar as shown in Figure 5.2. In order to infer this mapping from a certain shape or group, we either require a fixed set of shapes and groups with a fixed mapping, which we can automatically recognize, or a set of mappings the user can assign to the shapes. The first case has the advantage of allowing the user to directly employ a shape or group without having to assign an interaction behaviour to it but would limit him/her to the fixed set of known shapes. This observation is closely related to the concept of affordance, introduced by Donald Norman [131], which describes the interaction possibilities a human perceives when encountering an object or another thing in his/her environment, e.g. that one can throw a ball or push a button. We are specifically interested in which affordances different paper shapes or groups have.

Our initial hypothesis was that there would be a common ground on the affordances of certain shapes among users. To validate this hypothesis, we carried out an elicitation study to investigate which shapes users would cut out for specific use cases. We invited 20 participants (seven female, average age: 29.1 years, ranging from 15 to 69 years) and asked them to imagine different scenarios in which they had to control a set of appliances using paper shapes they cut out themselves as controls. We included the four scenarios listed below with the corresponding appliances or functionality which had to be controlled.

1. **Office:** blinds (up/down), temperature, ventilation, a light bulb (on/off), an LED with adjustable brightness and colour temperature, a doorbell.

2. **Movie remote control:** play/pause, fast-forward, rewind, volume, back button, ok-button, directions for selection cursor.

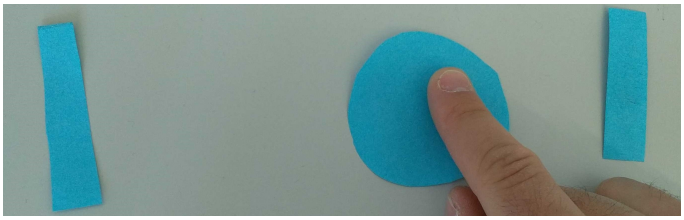


Figure 5.2: An example for a paper interface: a trackbar.

3. **Drone:** 3D movement (translation in each direction), rotation (horizontal rotation), taking a picture.

4. **Car:** air temperature, driver's seat temperature, driving modes (eco, comfort, sport), driver's seat adjustment (backrest tilt, seat pitch, seat height).

We provided the participants with sheets of paper in different colours and a pair of scissors, and let them cut out the shapes and groups they believed were best suited to provide the specific control functionality.

In contrast to our assumption, the shapes and assembled groups the participants created varied a lot. While some interfaces were implemented similar to our expectations (and to common everyday interfaces), many of them would not be understandable for another person without further explanation (cf. Figure 5.3). Often, the interfaces were relatively complex because a single shape would be assigned a range of different functions. We furthermore did not find a consistent use of differently coloured paper.

What we did find are common abstractions of interaction elements (or interaction patterns). The interaction elements may not have the same appearance (i.e., shape), but the same behaviour. For example, many participants use buttons which could be tapped and which all related to the same underlying behaviour despite being different in appearance. Hence, we decided to not implement an automatic recognition of shapes together with an automatic assignment of the interaction behaviour, but to implement a system which allows the user to create an arbitrary shape and assign the intended behaviour him-/herself from this set of identified common abstractions. The fact that all participants thought that their interface made sense concerning the given task emphasizes the strength of the

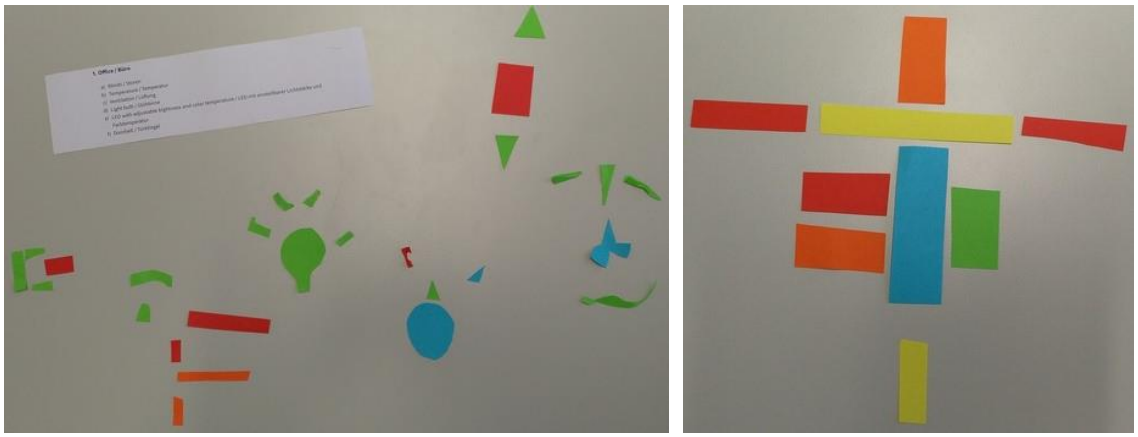


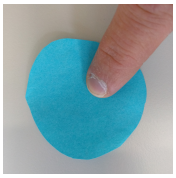
Figure 5.3: Two examples of interfaces participants designed. Left: An interface for controlling a smart workplace, similar to what one might expect. Right: A complex interface for controlling a movie.

underlying idea, i.e., that the interfaces can be fully personalized to each individual's preferences.

5.3.1 Touchets

From the elicitation study, we inferred a set of 25 abstract interactions, which we refer to as *touchets*. A *touchet* represents a behaviour a certain shape or a group of shapes should exhibit, e.g. a "finger slider" should allow a piece of paper to become a slider manipulated by the movement of the finger on top of it – note that this concept is *independent* of what the shape actually looks like. We found five categories of touchets, the two largest ones being buttons and sliders. In the following, we will shortly describe each identified touchet. In our implementation, touchets propagate their manipulation by returning events, which we also explain here. For some of the touchets, we added a picture displaying some elements that occurred in the study with visualizations showing how a shape may be moved or touched. However, these are merely examples of shapes or groups that could be assigned the corresponding touchet.

Button Touchets



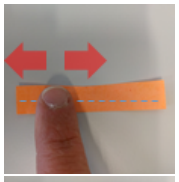
1. **Button (B):** A simple button which returns a "touched" event when the user's finger touches it.
2. **Hold Button (HB):** A button which can distinguish between a short and a long press. It also returns an event with the duration of being pressed.

3. **Positional Button (PB):** A positional button may contain several clickable zones, which are defined by the user.
4. **Positional Hold Button (PHB):** The same as a positional button, but with hold buttons instead of buttons.

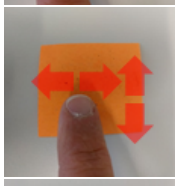
Finger Sliders

Finger sliders react to a user moving his/her finger above them, i.e. they track the position of the finger relative to the slider. They return this position as a relative value whenever

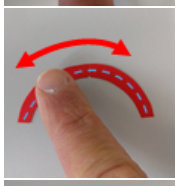
the finger is moved. The finger is always kept close to the slider.



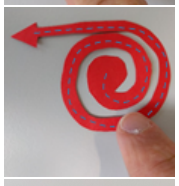
5. **Finger Slider (FS):** A 1D finger slider with a straight axis.



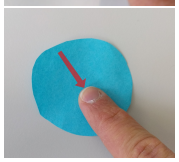
6. **2D Finger Slider (2FS):** A 2D finger slider that reacts to the movement of the finger in two directions.



7. **Curved Finger Slider (CFS):** A 1D finger slider, however, with a curved axis.



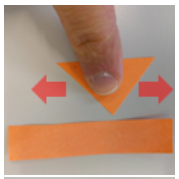
8. **Special Finger Slider (SFS):** A finger slider with an arbitrary shape.



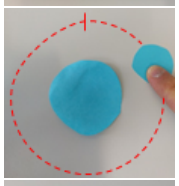
9. **Swipe Button (SB):** A finger slider that returns the direction of the swipe.

Shape Sliders

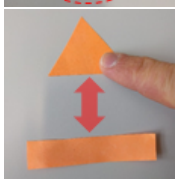
Shape sliders are similar to finger sliders with the difference that the value is not set by the finger directly, but by moving an indicator shape relative to a reference shape.



10. **Slider (S):** A 1D slider with an indicator shape and a reference shape. It returns the relative position of the indicator to the main axis of the reference shape, which is defined by the user by identifying the two endpoints of the axis.

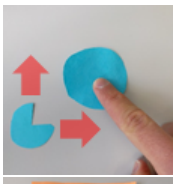


11. **Circular Slider (CS):** For circular sliders, the indicator shape can be rotated around the reference shape. The touchet returns the current rotation angle.



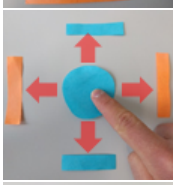
12. **Unbounded Slider (US):** A 1D slider without an upper bound. The returned value is the distance of the indicator shape to the reference.

13. **2D Unbounded Slider (2US):** An unbounded slider that returns the distances from the reference shape on two orthogonal axes.

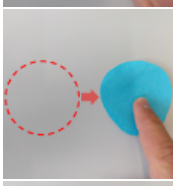


14. **Trackbar (T):** A trackbar consists of three shapes, two representing the lower and upper bounds, and the third being the indicator. It returns the relative position w.r.t. the bounds.

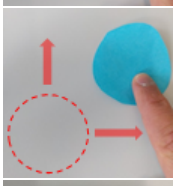
15. **2D Trackbar (2T):** A trackbar where the indicator can be moved on a 2D area bounded horizontally and vertically by four reference shapes. It returns the corresponding relative position for both axes.



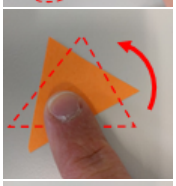
16. **Initial Position Offset Slider (IS):** A single shape that returns the current distance from the position it was first touched along a single axis.



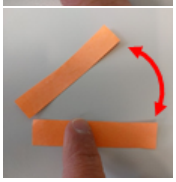
17. **2D Initial Position Offset Slider (2IS):** The same as above only returning the distance from the initial position on two orthogonal axes.



18. **Rotation (R):** A single shape that reports the current rotation angle relative to the position it was in at the beginning of being touched and relative to its initial position on the table.



19. **Angle (A):** Returns the current angle between two shapes.

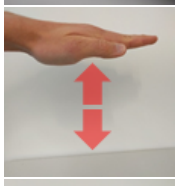


Hand Touchets

These interactions all concern the movement of the user's hand directly, and independent of the paper shapes. Hence, they do not fit into our concept of interacting with paper shapes and were also relatively rare. However, we list them here for completeness.

20. **Hand Horizontal Position (HP):** Returns the 2D position of the hand in the plane parallel to the interaction surface.

21. **Hand Rotation (HR):** Returns the rotation of the hand in the plane parallel to the surface around the center of the hand.



22. **Hand Height (HH):** Returns the height of the hand from the surface.

23. **Hand Tilt (HT):** Returns the two angles of a virtual plane represented by the hand relative to the surface.

Special touchets

24. **Existence (E):** The existence touchet consists of a single shape which returns a boolean value referring to whether it is currently placed on the surface or not.

25. **Containment (C):** Consisting of two shapes, it returns a boolean value referring to whether the smaller shape is currently contained within the larger one or not.

Touchets provide a powerful abstraction from the actual shapes and thereby enable users to personalize their TUIs. In addition, it is possible to assign multiple touchets to the same shape or group, which will then react to more interactions. Hence, our touchets approach allows users to model and create complex interfaces. Besides, this approach has the benefit that we do not have to classify the shapes, but only track each shape and group and its touchet assignment.

5.3.2 Frequency of Touchet Occurrence

The touchets presented above cover the complete set of interaction abstractions observed in the study. When using our approach, one has to assign the touchet functionality to the cut-out paper shapes. Providing the user with a large set of touchets might lead to

confusion and make it difficult to choose the appropriate touchet. Hence, we counted the frequency of occurrence of touchet instances in the study in order to select the most popular touchets for our prototype system. In total, we found 702 touchet instances. Figure 5.4 shows the frequency of each touchet. Only nine touchets account for over 90% of occurrences. Based on the frequencies, we decided to include all touchets with a count of at least five in our implementation. However, we excluded the "hand height" (HH), as it does not fit into our concept of paper interaction. Another touchet we omit is "existence" (E), as shapes may not have a unique appearance and it hence is difficult to track a shape visually which is taken away from the surface and reused later. The 13 selected touchets cover over 92% of occurrences; we hence provide a nearly complete set of interaction elements without overwhelming the user.

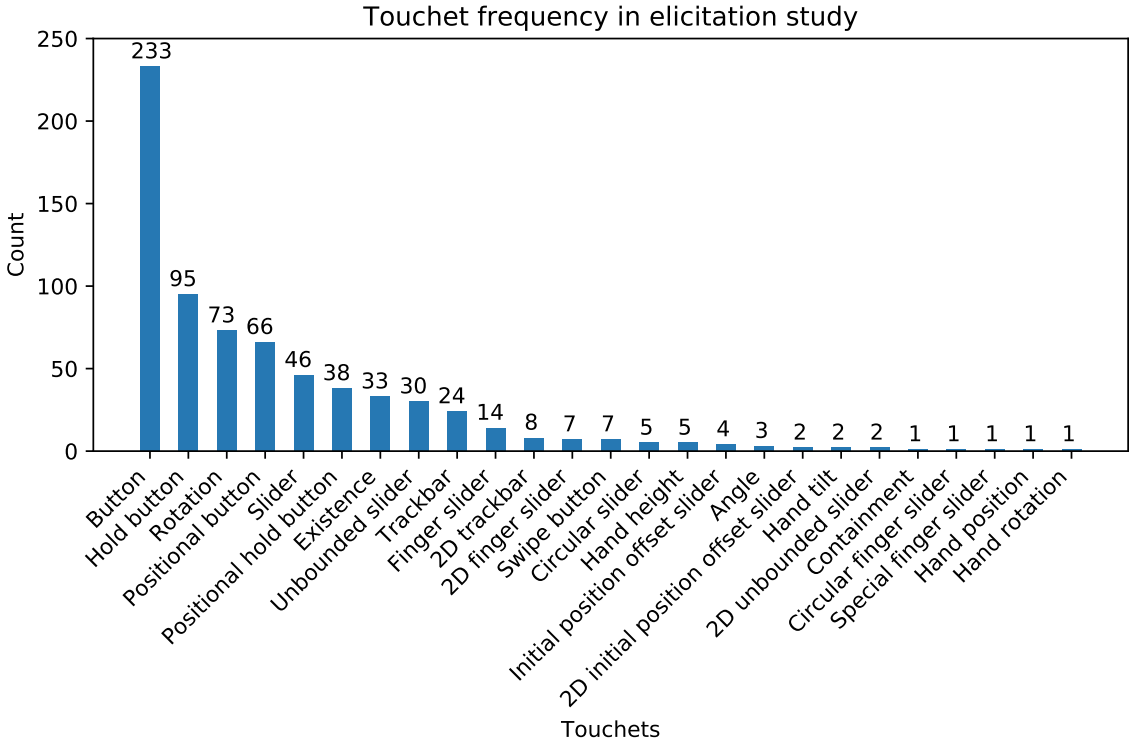


Figure 5.4: The frequency of occurrence of each touchet in the elicitation study sorted according to the frequency.

5.4 Tailored Controls

5.4.1 Method Overview

We first describe the typical process of using user-defined paper snippets for controlling appliances. The use of our prototype is furthermore showcased in a video available online [213]. After the user has cut out and arranged his/her shapes, he/she has to assign touchets to the shapes to add functionality, i.e. responsive behaviour, to them. This is done by placing the finger in the corner (shown as a blue square in Figure 5.1b), which opens a menu to select a touchet. For some touchets, the user may have to select several shapes, e.g. for a slider, an indicator and a referential element are required. Similarly, one can assign predefined actions to the shapes by opening another menu using the green square. Actions are tags which can be used by the attached application to distinguish which shape was touched (as there might be several buttons, for example). From now on the touchet instance emits the corresponding event carrying the relevant information about its state and the action tag whenever it is manipulated by the user. These events can then be used in any connected application, as explained below. Our implementation reacts to all changes immediately, e.g. movements of the reference shapes of a trackbar touchet instance change the corresponding bounds, which directly also changes the trackbar's value.

From the touchets we selected for our prototype, we can derive two main requirements: tracking the shapes and groups which have touchets assigned to them, and tracking the 3D-position of the hand, especially of the fingers. Here, we restrict the finger tracking to a single stretched out finger. A way to fulfil these requirements using only a single hardware sensor is to use an RGBD camera mounted above the interaction surface (cf. Figure 5.1a). The depth information is particularly important to detect when the finger

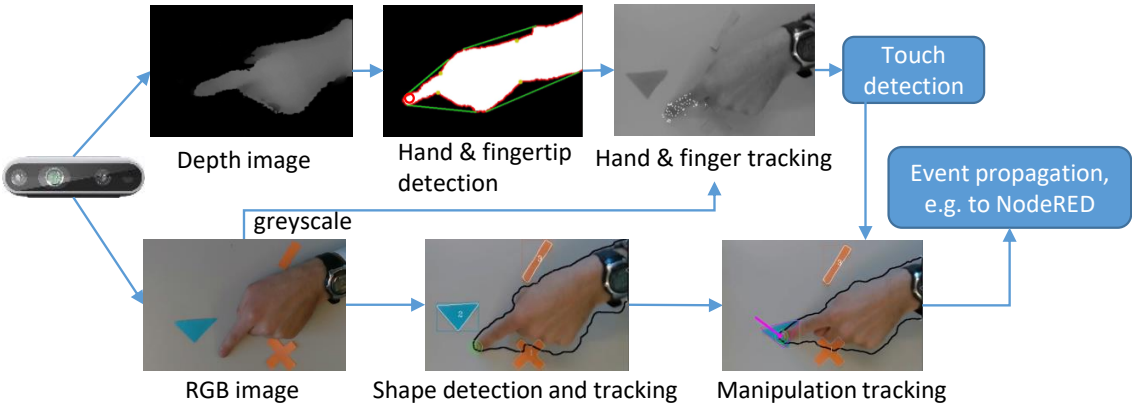


Figure 5.5: Our processing pipeline mainly consisting of finger and shape tracking based on the images from the depth camera.

touches the surface or a shape. An overview of our processing pipeline is depicted in Figure 5.5. On the depth image, we perform a colour-independent hand segmentation based on depth-thresholding. In the hand segment, we find an initial estimate of a stretched out finger. We then track the fingertip using an optical flow algorithm, which allows us to obtain the changes in the position of key points on the fingertip between each pair of consecutive frames. Knowing the 2D-position of the fingertip, we can obtain the finger's height from the depth image as well and thereby deduce whether the user is currently touching the surface or a shape on the surface. Based on the RGB image, we track all the shapes in the field of view of the camera. As soon as a shape is placed on the surface, it is assigned a unique identifier. Each shape is tracked continuously by a shape tracking algorithm we designed for this purpose, which returns an estimate of the translation and rotation a shape has undergone between two frames. The tracking is a challenging problem, as a significant part of a shape may be occluded by the user's hand. Finally, we combine the information about the finger and shape positions and thereby can detect whether the user is touching a shape, swiping over it, or moving it. If any of the actions performed matches the touchet specification assigned to the touched shape, an event is forwarded to a server, from where this information can be relayed to other applications. For example, a "button" touchet will not react to rotation, but a "rotation" touchet will.

For configuring the propagation of events, we use Node-RED [227], a programming tool which allows connecting and programming hardware devices, APIs, and online services in a GUI editor. Using Node-RED, users can completely customize their own application. For frequent use, the Node-RED parts can certainly also be pre-implemented, so that the user can simply connect the touchet events to these pre-configured actions.

5.4.2 Set-up

For the RGBD camera, we use an Intel RealSense D435, which we mount 40 cm above a table, as shown in Figure 5.1a. The camera monitors a rectangular area of about 43 cm by 32 cm. The camera stream is processed by a desktop computer with an Intel i7-4790K 4x4 GHz CPU 16 GB of RAM. All the processing runs on the CPU. The source code of the system can be downloaded from GitHub [212].

5.4.3 Image Preprocessing

Before detecting the hands and shapes, we first preprocess the colour and depth images from the camera. The RGB image is merely aligned with the depth image so that the coordinate systems correspond to each other. The depth image requires further processing

due to noise: We first apply a temporal filter to reduce the noise. Additionally, to remove spurious errors in the depth image where the values are clearly too high, we employ a hole-filling algorithm, which takes the neighbouring pixels of such a hole into account. Both algorithms are available in the RealSense library. Another problem is that the camera plane is usually not perfectly parallel to the interaction surface, resulting in some areas of the surface being perceived as further away, which interferes with the hand and finger detection on the surface. In order to prevent this from happening, we take a snapshot of the depth values when starting the system and henceforth subtract these from the currently recorded image. Consequently, all points on the surface have the same depth, which facilitates further processing.

5.4.4 Finger Detection and Tracking

One primary requirement is the ability to track the hand and fingers in order to detect when the user is touching and manipulating the shapes. In an optimal case, we would know the pose of the complete hand. For this purpose, we tested several hand pose estimation algorithms such as [132], but none showed a satisfying accuracy although having high computational requirements. Hence, we restricted the way users may manipulate shapes to the most common one, touching them with a single fingertip, and designed an efficient tracking algorithm for this case. Furthermore, we require the contours of the hand to know when shapes are occluded. We assume that there are no objects in the field of view of the camera apart from the paper shapes and the hand. Hence, we can distinguish the hand by applying thresholding to the depth image, as the shapes are flat. We threshold the image at a level of 1 mm to obtain a binary mask of the hand and apply a contour finding algorithm. From all the resulting contours, we take the largest one, as we assume the hand and arm to be the only larger 3D object in the scene. Thereby, we obtain a rough estimate of the hand and arm position. To find the fingers, we search for defects in the convex hull around the hand. The fingertips are the corners of the convex polygon. Unfortunately, even after preprocessing the depth image, there is still a significant amount of noise, which deteriorates the accuracy of the fingertip position. Hence, we do not only search for the fingertip in the 1 mm-thresholded binary mask, but also in higher masks. We thereby obtain several corner points along the top of the fingers, as illustrated in Figure 5.6. Afterwards, we apply a closing morphology on the corner points in order to merge nearby points. Fingertips then appear as the ends of point clusters.

We assume that the relevant fingertip for touching is the point furthest away from the arm. We can find the arm as the part of the body which intersects the image frame.

The detection step described above by itself merely provides the current location of

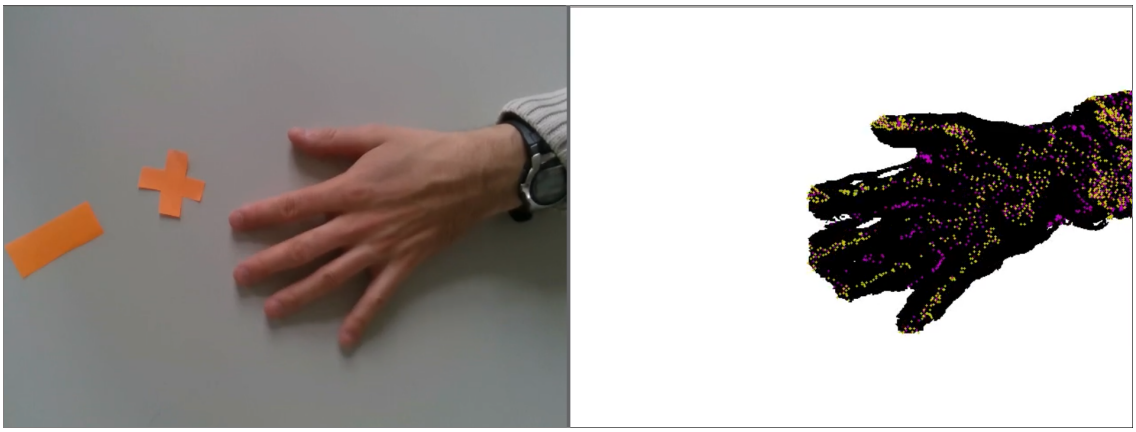


Figure 5.6: The resulting corner points of the finger finding algorithm applied on different depth levels.

the fingertip, but not its movement over time, which is relevant to recognize and track interactions with the shapes. In order to actually track the finger, we apply the Lucas-Kanade optical flow algorithm [118] to the detected finger region in the greyscaled RGB stream. The algorithm is able to find salient key points in the region and track their position across multiple frames. We apply it as soon as a finger is detected in the images. As multiple points are tracked, we can not only calculate the translation of the finger across frames but also its rotation by matching the two corresponding point sets from two consecutive frames. This is done by computing the best-fitting rigid transformation [158]. With the 2D location of the fingertip, we can directly extract its height above the surface from the depth map and deduce whether the user is touching a shape or not.

Evaluation of Fingertip Detection In order to evaluate the accuracy of the fingertip detection, we recorded several sequences during which a participant moved his/her fingers below the camera. We then manually annotated the position of the fingertip to compare it to our estimation. In total, we annotated 1137 frames from six participants (note that our detection method works independently of skin colour). On average, the error was 15 pixels which corresponds to approximately 1 cm. However, this average was influenced by a few large errors which occur when the hand is entering the field of view of the camera. The median error was 8.6 pixels, which is accurate enough given shape sizes of a few centimetres.

5.4.5 Shape Tracking

Detecting and tracking the shapes is performed purely on the RGB image, as the depth image does not contain any information about the flat paper shapes. We assume the interaction surface to be unicoloured with low saturation, e.g. a standard table. To detect the shapes, we convert the RGB image to the HSV colour space and threshold the saturation channel (assuming the shapes have a different colour than the table). In the resulting binary mask, we find the contours. From these shape candidates, we exclude those in the area covered by the hand contour, as illustrated in Figure 5.7. Whenever a shape is occluded by the hand or arm, we try to match reappearing shapes by their position to recover the shape identity.

5.4.6 Tracking Manipulations of Shapes

Knowing the position of the fingertip and the shapes, we can detect when a user touches a shape. To fulfil the necessary requirements to recognize all possible touchet interactions, we have to detect for how long a shape is touched and whether it is swiped upon, as well as the translation and rotation of the shape in case of a movement. A swipe can be distinguished from moving the shape by the fact that only the finger, but not the shape itself moves. For movements of shapes, we track the change in translation and rotation from a shape's appearance for every frame since the shape was touched as this might induce a parameter value change which we want to be able to follow continuously. Since the shape is occluded by the finger (and the finger position might also shift during the movement), applying a direct transformation calculation through a best-fit algorithm as for the finger tracking is not possible. Instead, we take a snapshot of the shape at the moment it is first touched and calculate the translation between the currently detected shape and the

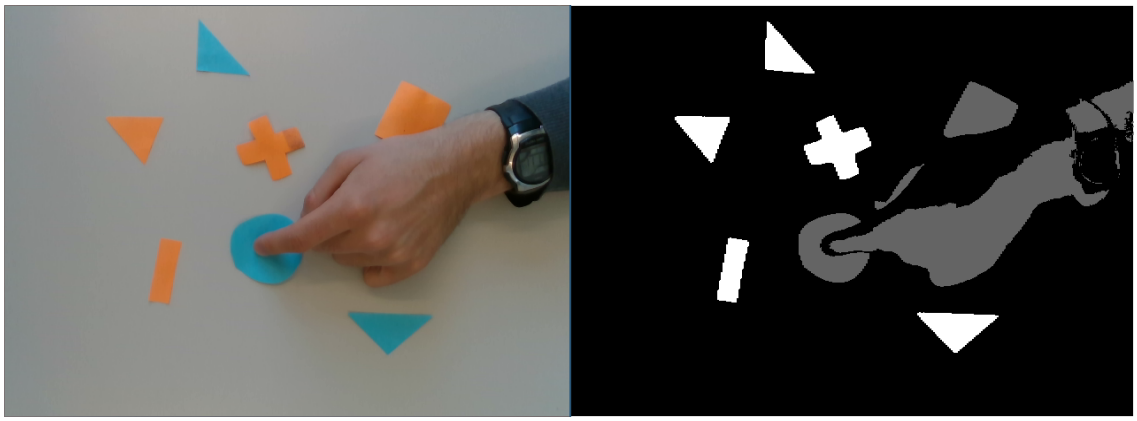


Figure 5.7: The binary shape masks with the detected hand region.

snapshot. For the rotation, we iteratively test for the best fit employing an efficient ternary search. The search uses the rotation estimate of the finger as a starting point. However, as the finger position and rotation might change independently of the shape during the movement, this can only be used as a first indication. The measure for a fit is the area of intersection between the detected shape and the snapshot. Note that we can also track the rotation of circular shapes, as the finger partly occludes a circle during interaction resulting in a perceived shape that has a "hole" in the location of the finger, which allows the tracking of the rotation.

Evaluation of Shape Tracking To evaluate the shape tracking accuracy, we recorded several sequences when moving a squared shape at different speeds, also including rotations. For 1,529 frames, we manually annotated the positions of the four corners of the square and compared them to the estimate calculated by our tracking algorithm. We calculated the average displacement of a corner point. The mean error over all the frames was 6 pixels (0.4 cm), which is accurate enough for our purpose.

5.5 User Study

To evaluate the entire process of creating a TUI with our prototype, we invited six new participants to a second study (two females, 18 to 28 years old). As we planned to let the participants build a complete application from scratch, including the connection to and implementation of the Node-RED flow, we only invited computer science students with programming knowledge. We asked them to implement interfaces and the Node-RED flows for the three following applications:

1. Driving a little toy rover, as shown in Figure 5.1c, which can drive forwards, backwards, and rotate to the left and right.

2. Creating a music playback control based on an `MPD` [229] wrapper we provided. The interface should control volume, play and pause, and switching to the next or previous song.

3. Creating an interface for controlling a slide show. In order to do so, we provided a wrapper in Node-RED to access the `xdotool` [239], which allows emulating keyboard presses. The interface should allow starting and stopping a slide show, as well as to move to the next and previous slides.

The order of implementing each of these applications was different for each participant. Three days before the study, we gave the participants basic information on the required

components and asked them to get familiar with `MPD` and `xdotool`. During the study, we provided the participants with an overview of the available touchets besides the task descriptions. We set no time limit and let the participants try any interface they wanted. After the study, we asked them several questions concerning our prototype and let them fill in a User Experience Questionnaire (UEQ) [110] score sheet to obtain a standardized measure of the user experience (the questionnaire is available online [225]). The UEQ includes 26 items on a seven-point score ranging from -3 to 3. The output of the UEQ analysis is a set of scores in the following six dimensions: attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty.

All participants successfully completed the tasks. On average, a participant spent 2 hours and 18 minutes on all three tasks. Note that this includes all necessary parts, i.e., the design of the paper interface and the assignment of the touchets, as well as the Node-RED implementation. As participants spent most of the time on the Node-RED implementation, this latter part could also be pre-implemented for certain real-world applications, speeding up the use of our method significantly, e.g. for most of our tasks from a magnitude of half an hour to an hour down to a few minutes. As we let participants try out interfaces as they wished, they did not optimize for time. One participant enjoyed using the system and optimizing his interface so much that he stayed for five hours. On the other hand, another participant created an entire interface in 11 minutes. Furthermore, the average task time significantly decreased from 70 minutes for the first task to 32 minutes for the second.

The resulting scores for the six dimensions of the UEQ are shown in Figure 5.8. For all dimensions, the scores are positive, indicating that participants generally enjoyed using our prototype. Most noticeable are the scores for perspicuity, stimulation, and novelty, which imply that it was easy to learn and to understand, that the participants were motivated and excited to use the approach, and that they believed it to be a creative new method for designing interfaces. The worst characteristic is efficiency because our prototype cannot track fast movements of the finger due to significant motion blur in the depth stream, i.e., sometimes participants had to repeat their interactions. Moreover, several participants mentioned the relatively long time it takes to configure the necessary Node-RED flows, especially for inexperienced users. This most likely also influenced the attractiveness score. However, the Node-RED part is only used to forward the interaction events and could be replaced by a different mechanism, or the flows could be pre-implemented for common devices to be controlled.

As part of a survey that followed the study, participants confirmed that our method was easy to understand and that most of them enjoyed using the prototype, especially to try out different interfaces. They also suggested different scenarios in which the *Tailored Controls* could be used, such as for rapid prototyping, for interfaces in changing environments, or for controlling home devices.

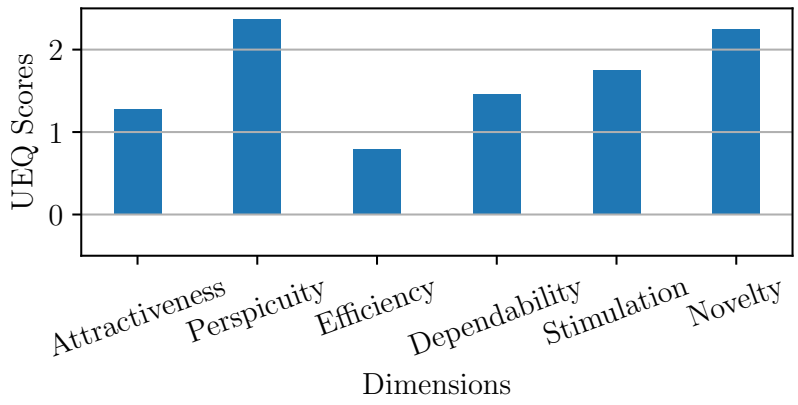


Figure 5.8: The average results of the UEQ.

5.6 Discussion

Our technical evaluation and the results from the presented user study show that we created a functioning prototype with appropriate performance (e.g., regarding shape detection) which was positively received by participants. This is in spite of the fact that our prototype has several limitations, which are discussed below.

Despite these limitations, we were able to show that our suggested *touchet* taxonomy that we derived from the results of our elicitation study can indeed be used for the implementation of personalized TUIs. With our approach, we significantly progress beyond the state-of-the-art compared to other current systems that enable dynamic TUIs. In addition to not requiring fiducial markers, our system enables more natural interactions due to the explicit finger tracking functionality, and it provides users with a wide range of interaction abstractions to select from when creating TUIs.

We built our prototype with the intention to demonstrate the feasibility of our approach and evaluate it in the user study. However, we believe there is a multitude of different application scenarios where a system such as ours could prove useful as already mentioned in the introduction.

First, our approach can be used for the rapid prototyping of functional TUIs – paper prototyping is a common technique for interface design. However, such interfaces are usually not functional, and the effects of using them have to be imagined by prototypers. With our system, it is possible to immediately test and experience functional interfaces, which we believe would spur creativity. This is backed by our user study, where participants were very motivated to try out different interaction elements, in particular, because our system does not restrict users to employ predefined generic paper shapes. Consequently, *Tailored Controls* can bridge the gap between the “ideation” and “implementation” stages

in the design thinking process [39], enabling the designer to be more creative and faster by providing direct feedback on how a functionally implemented interface reacts.

Second, our system can be used in the context of dynamically changing workspaces, e.g., to provide ad-hoc interfaces to devices that are used only infrequently – we imagine this to be of use especially in crowded workspaces (e.g., in industrial workshops), where it allows users to carry interfaces with them and deploy them tactically, rather than attaching TUIs to the machines they intend to control. To this end, we want to emphasize that the presented system and our prototype is not limited to paper alone, but can be used with other (sufficiently flat) materials.

Third, our approach enables a very high degree of personalization in terms of the TUIs that can be created by users. It thus enables individualized user interfaces, for instance in the context of accessibility constraints: with our approach, it is possible to create interfaces that are optimally suited for individual disabled persons, given their specific type of disability and context constraints.

5.6.1 Limitations and Future Work

Our approach has several limitations, both from a theoretical and a practical perspective, which we intend to approach in the future. Our current implementation of the finger tracking algorithm only allows the use of a single finger for manipulations, and there is no multi-touch support. Furthermore, it is difficult to track fast movements of the fingertip, as fast movements create sequences of blurry images in the depth stream. Finally, the interaction space is currently limited to a relatively small flat area. We envision that these problems can be solved in the future by better hand pose estimation algorithms on the one hand and by improved hardware on the other. Our approach could directly be adapted to such improvements. A further issue is that our system requires an RGBD camera to be available to monitor the scene, i.e. the approach is most applicable for fixed workspaces. With future improvements, we, however, expect the area which may be covered to grow.

On a philosophical level, the use of flat paper shapes may no be considered tangible in a strict sense. However, we view tangibility as the physical, material presence of the elements used, i.e. that they can be touched, moved, or reconfigured without interacting with a digital proxy device that displays the user interface. We believe already this property of the paper shapes is beneficial for the interaction process, as it enables the interactions to take place in the real world, an environment the user is naturally accustomed to. In terms of taking advantage of the full properties of paper, one could experiment with other forms paper interfaces may take, such as folding or crushing the paper, which would further enhance the tangible experience. Nevertheless, none of our participants in the

elicitation produced any of these interfaces, neither did anyone mention that this could be a possibility, and we did not restrict them only to cut out paper shapes. Furthermore, one could include different types of paper or materials, such as fabric, which is an interesting aspect of future work. We partly already included this in the study by providing differently coloured paper. However, this property was rarely incorporated in the interface designs by the participants. Further, we do believe our approach could naturally be used with other (flat) materials as well, such as fabric or even play dough, which would enable an even easier reconfiguration of interfaces. Other materials could also solve practical environmental issues paper shapes might have; for example, they are easily blown away by wind or a draft.

5.7 Conclusion

We presented the *Tailored Controls* approach that enables the simple creation of personalized TUIs that are made from plain paper but can be connected to virtually any application. In a first user study with 20 participants, we found 25 interaction abstractions implicitly employed by the participants. We built a functional prototype and implemented 13 of the most common of these abstractions. Our system tracks a user's fingertip as well as the paper shapes and is thereby able to detect interaction events, including tapping, rotating, swiping, sliding, and many more. Our prototype allows the simple creation of user interfaces for diverse applications, as we proved in a second user study, and demonstrates the feasibility of our initial concept. By using a set of abstractions that covers a wide range of potential interaction elements, we are able to give users the freedom to create the paper shapes they want and afterwards add corresponding functionality themselves, instead of having to choose from a predefined set of generic elements. The only hardware required is an RGBD camera that is mounted above the interaction surface.

Tailored Controls demonstrates the feasibility of inexpensive reconfigurable TUIs, and one main contribution is to enable further research in the direction of customizable personalized TUIs. Immediate applications of our approach include the rapid prototyping of functional TUIs, the creation of tactical interfaces for sporadic interactions, and the generation of individualized TUIs that are optimal for concrete individual contexts, for instance for the benefit of disabled persons. In addition, triggered by the expected proliferation of RGBD cameras, we expect our approach to enable novel interaction methods in domestic and public environments, as well as in mobile scenarios.

Chapter 6

Conclusions

In this chapter, we briefly summarize the contributions of this dissertation and further discuss remaining limitations and challenges for future work concerning the interaction with smart devices. We also propose different paths for future research in the field of ubiquitous computing that we find interesting.

6.1 Summary of Contributions

The amount of smart devices in our environment is continuously growing as we use more digital devices in general but also, more and more everyday devices are becoming smart. This growth impels the exploration and development of enhanced ways of interaction between humans and smart devices, as a natural and intuitive interaction constitutes one of the primary quality features of device functionality. This goal established the overarching theme of this dissertation. We presented different methods of how we could enable or facilitate the interaction of humans with smart devices, primarily focusing on how the user could provide input. First, we presented two gesture recognition methods, *TouchSense* and *GestEar*, based on different modalities, including electromyography, motion, and sound. We continued by examining how interactions could take place in the AR space by representing the user interface by AR widgets. Within that space, we also investigated different interaction methods. Finally, we proposed a method which allows users to build their own user interfaces from paper.

The first project, *TouchSense*, built on the idea that identifying the finger used for touching and measuring the force of the touch provides valuable information on manual interactions. This information can be inferred from electromyography (EMG) of the fore-

arm, measuring the activation of the muscles controlling the hand and fingers. TouchSense classifies the finger touches using a novel neural network architecture and estimates their force directly on a smartphone in real time based on data recorded from the sensors of an inexpensive and wireless EMG armband. Using data collected from 18 participants with force ground truth, we evaluated our method's performance and limitations. Our approach could allow for new interaction paradigms with appliances and objects, which we exemplarily showcased in four applications.

In Chapter 3, we introduced *GestEar*, a gesture recognition method for sound-emitting gestures, such as snapping, knocking, or clapping, using only a simple smartwatch. The technique exploits motion information from the built-in accelerometer and gyroscope, as well as audio data recorded by the smartwatch microphone as input. We designed a convolutional neural network architecture for gesture recognition, designed to run locally on resource-constrained devices such as standard smartwatches, which achieves a user-independent recognition accuracy of 97.2%. We further showed how to incorporate gesture detection and gesture classification in the same network, compared different network designs, and showcased several applications built with our method. Our evaluation revealed that our system benefits from both the sound input as well as from the IMU data. Our method could easily be added as a software update and thereby extend the functionality of wrist-worn devices which contain an IMU and a microphone, such as smartwatches and potentially also fitness trackers etc.

After presenting two gesture recognition methods, we introduced a concept which also includes the recognition and selection of smart devices in Chapter 4. After the user selects a device, the device's user interface is presented in augmented reality. We investigated different interaction methods from an ego-centric perspective. We examined how users can control appliances through augmented reality directly. The physical objects are augmented with interaction widgets, which are generated on demand and represent the connected devices along with their adjustable parameters. We explored three ways of interacting with the virtual widgets: (1) in-air finger pinching and sliding, (2) whole-arm gestures rotating and waving, (3) incorporating physical objects from the surroundings and mapping their movements to the interaction primitives. We compared these methods in a user study with 25 participants and found significant differences between them in terms of user preference, speed, and granularity. We also developed a second method to allow users to create and modify logical control connections between multiple devices in AR. This approach facilitates the understanding of existing connections and their modification.

In the contributions mentioned above, we created different interfaces for the user. In the final project *Tailored Controls*, presented in Chapter 5, we provided a framework to allow the user to build his/her own tangible interface from materials as simple as paper. Users cut out shapes and assemble them to their personalized interfaces. They can assign control

functions to these paper shapes via a simple configuration interface. Our framework then recognizes interactions with the paper interfaces and forwards these events to any connected application. Our system is based on markerless tracking of the user's fingers and the paper shapes on a surface using an RGBD camera mounted above the interaction space, which is the only hardware sensor required. Our approach and system are backed up by two studies where we determined what shapes and interaction abstractions users prefer and verified that users can indeed employ our system to build real applications with paper snippet interfaces. *Tailored Controls* could not only enable an improved paper prototyping process in the future but also make it possible to have personalized workspace interfaces for workers, or potentially even help people with impairments, such as blindness, to configure a personalized interface for themselves.

6.2 Demonstrators and Code

Besides designing algorithms for the problems mentioned above we also implemented each method, mostly on wearable and resource-constrained devices, in order to demonstrate the methods in different real-world scenarios and prove that it is possible to run them even on current off-the-shelf hardware. We also created demonstration videos showing our prototypes alongside the papers which are available at the URL references given in Table 6.1. The table also provides references linking to the source code which we published for some of the projects.

Projects	Chapter	Video reference	Code reference
TouchSense	2	[215]	[214]
GestEar	3	[210]	[209]
AR interactions	4	[211]	-
Tailored Controls	5	[213]	[212]

Table 6.1: The URLs to the video and source code resources for each project.

6.3 Open Problems and Future Work

As mentioned in Chapter 1, we are aware of several limitations of our proposed solutions, which we want to discuss here and which may direct the course of future work. First of all, there remain technical challenges such as energy consumption when our methods run on mobile devices. Second, there is also future work to be done in order to achieve

a complete solution for the interaction with smart devices, as our work mainly focuses on the aspect of providing input to such systems, and not how interaction is initiated, for example, which would include the selection of the smart device.

6.3.1 Energy Consumption and Battery

While we did focus on optimizing our methods to run in real time also on wearable, resource-constrained devices, we acknowledge that further work is necessary to also optimize our methods for battery lifetime, especially in the cases of *TouchSense* and *GestEar*. Both methods require far less than the full processing capacity of the main processor. However, keeping the main processor awake at all times for gesture recognition is very energy-intensive. Nevertheless, several hardware manufacturers are already working on special low-power chips, which allow algorithms to run without activating the main processor. Furthermore, these chips even make it possible to bypass the main processor for sensor processing by delivering the sensor data directly to the special-purpose chip. Examples of such developments are Google's CHRE (Context Hub Runtime Environment [222]) co-processor platform or Qualcomm's Neural Processing SDK [237]. Qualcomm's SDK gives developers access to specialized engines for running neural networks or digital signal processors for low-power (but fast) processing. While the processing capabilities currently restrain us from running a larger classification model on low-power co-processors (e.g. CHRE only allows so-called nano-apps with very low processing requirements), one could still run smaller detection models, which wake up the main processor for enhanced processing only in the few cases a relevant event has most likely happened. These hardware improvements could enable the continuous use of methods such as the ones proposed by us.

6.3.2 Further Miniaturization and Hardware Improvements

While GestEar already runs on a smartwatch and hence would be socially acceptable, some of our other methods require specialized hardware not yet ready for everyday use. TouchSense requires an EMG armband and our AR prototype a HoloLens. Users will not be willing to wear either piece of equipment over extended periods of time, due to the current weight and form factor. Nevertheless, we believe that such devices will be improved rapidly, the EMG armband might be sewn into a sleeve, and AR glasses might soon have the form factor of usual glasses as can be anticipated from devices such as the North focals [234]. Along with these specific improvements, we will experience further miniaturization, but also enhancement of hardware, which will allow us to include our methods into smaller devices, which are less obtrusive to the user.

6.3.3 Building a Complete Interaction System

As mentioned in the introduction, in this dissertation we mainly focus on the problem of how a human can provide input in an interaction with a smart device, in particular concerning our gesture recognition projects *TouchSense* and *GestEar*. As depicted in Figure 6.1, we can conceptually view an interaction with a smart device as a process consisting of three stages. First, recognizing and selecting the smart device, then allowing the human to provide input, e.g. through gestures, and finally interpreting the human's input and determining which device command is the most appropriate according to the current device state and the input.

Our two mentioned methods provide solutions for the second stage. Future research would have to consider combining our methods with solutions for the other stages to form a complete interaction system.

The first stage could be solved by different mechanisms, such as visual recognition of the smart devices, attaching visual markers to the devices, or installing an infrared transmission system. Visual recognition purely based on the appearance of the devices has the disadvantage that two devices with the same appearance cannot be distinguished.

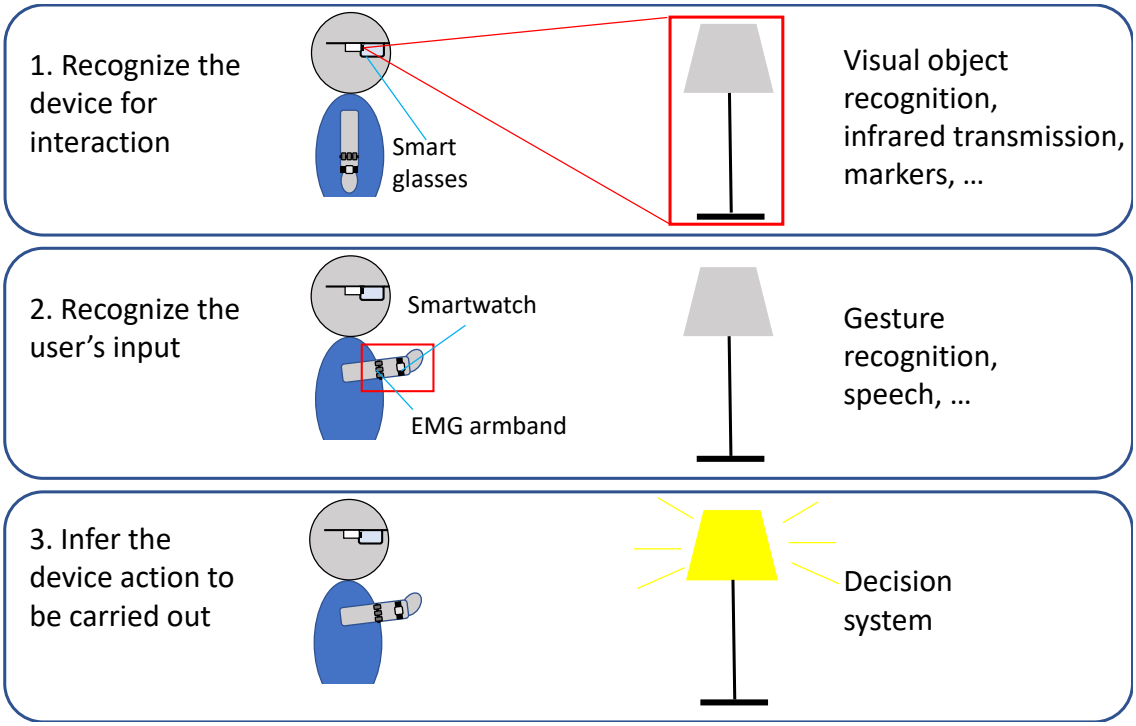


Figure 6.1: A concept of separating the process of interaction with a smart device into three stages.

However, the other methods require the augmentation of the environment, which results in a higher installation effort and reduced flexibility.

The third stage requires a decision system to estimate the most likely device action based on the state of the device and the command issued by the human, i.e. the method has to provide a translation from a human command to the most appropriate device action. Nowadays, this translation is usually simply hard-coded, as we also did in our demo applications. A solution could be achieved employing probabilistic models which return the most likely device command based on the combination of device state (which is known) and the human command determined in the second stage. These models could then also adapt to the human's preferences over time. Our work on representing user interfaces in AR provides an alternative solution to this challenge. There it is possible to show all interaction possibilities to the user and let him/her decide which action to carry out, i.e. a command translation would not be necessary any longer. Our last contribution, *Tailored Controls* also circumvents this issue by letting the user make all design decisions on the interface and its use him-/herself.

Another crucial challenge for the practical application is the combination of different modalities, such as gestures and speech. Humans usually do not use a single modality in isolation for communication. Hence, concerning the long-term goal of ubiquitous computing to make computers less apparent and provide a natural interaction environment, we should aim to find a meaningful way to combine all human means of communication.

Besides focusing on methods which are actively controlled by the user, such as issuing gestural or spoken commands, it is essential to further include methods which passively sense the human's context, both his/her internal state and the state of the environment. These could allow us to gain more information on the circumstances the user is in and under which conditions he/she is trying to complete a task and thereby assisting him/her in the best way possible. For example, we could potentially employ electroencephalography (EEG) or facial temperature measurements [105] to measure the human's workload and consequently adapt the presentation of information, such as reducing the complexity at a high mental workload. Or we could use eye tracking to infer where the user is directing his/her attention.

We believe that both the combination with other modalities and the incorporation of passive sensing are necessary to achieve the final objective of ubiquitous computing as envisioned by Mark Weiser and discussed in Chapter 1. These objectives open up many exciting opportunities for further research in the future.

6.4 Final Remarks

In this dissertation, we presented different methods enhancing the interaction with smart devices. Our contributions focus on different types of user interfaces: gestural interfaces, interfaces represented in augmented reality, and tangible interfaces, which the user may personalize. Each contribution advances the state-of-the-art in terms of performance and novelty of the solution. Overall, we believe we could deliver our share to the goal of creating more natural and intuitive interaction systems, which bridge the interaction gap between humans and smart devices. We furthermore hope that solutions such as ours lead to more accessible digital technology, which may benefit more people in the future. We are aware that further efforts are necessary, and many questions remain open to be solved in pursuance of this goal. Nevertheless, this research area provides opportunities for future research, as discussed in the previous section.

Bibliography

[1] S. A. Ahmad and P. H. Chappell. Surface EMG pattern analysis of the wrist muscles at different speeds of contraction. *Journal of Medical Engineering & Technology*, 33(5):376–385, 2009. PMID: 19440916. doi:10.1080/03091900802491246.

[2] B. A. Akyol, A. W. Jackson, R. Krishnan, D. Mankins, C. Partridge, N. Shectman, and G. D. Troxel. Smart office spaces. In *USENIX Workshop on Embedded Systems*, pages 29–31, 1999. URL: https://www.usenix.org/legacy/events/es99/full_papers/akyol/akyol.pdf.

[3] J. Al-Muhtadi, M. Anand, M. D. Mickunas, and R. Campbell. Secure smart homes using Jini and UIUC SESAME. In *Proceedings of the Annual Computer Security Applications Conference*, ACSAC '00, pages 77–85. IEEE, 2000. doi:10.1109/ACSAC.2000.898860.

[4] R. Albaghli and K. M. Anderson. A vision for heart rate health through wearables. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, pages 1101–1105, New York, NY, USA, 2016. ACM. doi:10.1145/2968219.2972715.

[5] A. Alkan and M. Günay. Identification of EMG signals using discriminant analysis and SVM classifier. *Expert Systems with Applications*, 39(1):44–47, 2012. doi:10.1016/j.eswa.2011.06.043.

[6] C. Amma, T. Krings, J. Böer, and T. Schultz. Advancing muscle-computer interfaces with high-density electromyography. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '15, pages 929–938, New York, NY, USA, 2015. ACM. doi:10.1145/2702123.2702501.

[7] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, J. Chen, J. Chen, Z. Chen, M. Chrzanowski, A. Coates, G. Diamos, K. Ding, N. Du, E. Elsen, J. Engel, W. Fang, L. Fan, C. Fougner, L. Gao, C. Gong, A. Hannun, T. Han, L. V. Johannes,

B. Jiang, C. Ju, B. Jun, P. LeGresley, L. Lin, J. Liu, Y. Liu, W. Li, X. Li, D. Ma, S. Narang, A. Ng, S. Ozair, Y. Peng, R. Prenger, S. Qian, Z. Quan, J. Raiman, V. Rao, S. Satheesh, D. Seetapun, S. Sengupta, K. Srinet, A. Sriram, H. Tang, L. Tang, C. Wang, J. Wang, K. Wang, Y. Wang, Z. Wang, Z. Wang, S. Wu, L. Wei, B. Xiao, W. Xie, Y. Xie, D. Yogatama, B. Yuan, J. Zhan, and Z. Zhu. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *Proceedings of the International Conference on Machine Learning*, ICML '16, pages 173–182. JMLR.org, 2016. URL: http://proceedings.mlr.press/v48/amodei16.pdf.

[8] M. Anand, J. Al-Muhtadi, M. D. Mickunas, and R. H. Campbell. Smart home: A peek in the future. Technical report, University of Illinois at Urbana-Champaign, 1999.

[9] D. Archer. Unspoken diversity: Cultural differences in gestures. *Qualitative Sociology*, 20(1):79–105, Mar 1997. doi:10.1023/A:1024716331692.

[10] J. W. Astington. *The child's discovery of the mind*. Harvard University Press, 1993.

[11] D. Avrahami, J. Wobbrock, and S. Izadi. Portico: Tangible interaction on and around a tablet. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '11, pages 347–356, New York, NY, USA, 2011. ACM. doi:10.1145/2047196.2047241.

[12] M. Bâce, V. Becker, C. Wang, and A. Bulling. Combining gaze estimation and optical flow for pursuits interaction. In *Symposium on Eye Tracking Research and Applications*, ETRA '20, New York, NY, USA, June 2020. ACM. doi:10.1145/3379155.3391315.

[13] M. Bâce, T. Leppänen, D. G. de Gomez, and A. R. Gomez. ubiGaze: Ubiquitous augmented reality messaging using gaze gestures. In *SIGGRAPH ASIA Mobile Graphics and Interactive Applications*, SA '16, pages 11:1–11:5, New York, NY, USA, 2016. ACM. doi:10.1145/2999508.2999530.

[14] M. Bâce, P. Schlattner, V. Becker, and G. Sörös. Facilitating object detection and recognition through eye gaze. In *Proceedings of the Workshop on Object Recognition for Input and Mobile Interaction at the International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, pages 1–3. ACM, 2017.

[15] M. Bâce, G. Sörös, S. Staal, and G. Corbellini. HandshakAR: Wearable augmented reality system for effortless information sharing. In *Proceedings of the Augmented Human International Conference*, AH '17, pages 34:1–34:5, New York, NY, USA, 2017. ACM. doi:10.1145/3041164.3041203.

[16] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, ICLR '15, 2015. URL: http://arxiv.org/abs/1409.0473.

[17] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio. End-to-end attention-based large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '16, pages 4945–4949, March 2016. doi:10.1109/ICASSP.2016.7472618.

[18] R. Barrett and P. P. Maglio. Informative things: How to attach information to the real world. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '98, pages 81–88, New York, NY, USA, 1998. ACM. doi:10.1145/288392.288576.

[19] T. Bartindale, D. Jackson, K. Ladha, S. Mellor, P. Olivier, and P. Wright. RedTag: Automatic content metadata capture for cameras. In *Proceedings of the International Conference on Interactive Experiences for TV and Online Video*, TVX '14, pages 19–22, New York, NY, USA, 2014. ACM. doi:10.1145/2602299.2602303.

[20] V. Becker. Adaptation of a cognitive decision-making model through the application of workload recognition, 2014. Bachelor's Thesis, Karlsruhe Institute of Technology.

[21] V. Becker. Augmented humans interacting with an augmented world. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '18, pages 428–429, New York, NY, USA, 2018. ACM. doi:10.1145/3236112.3236179.

[22] V. Becker, M. Bâce, and G. Sörös. Wearable machine learning for recognizing and controlling smart devices. In *Proceedings of the Workshop on Object Recognition for Input and Mobile Interaction at the International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, pages 1–3, New York, NY, USA, Sep 2017. ACM.

[23] V. Becker, L. Fessler, and G. Sörös. GestEar: Combining audio and motion sensing for gesture recognition on smartwatches. In *Proceedings of the International Symposium on Wearable Computers*, ISWC '19, pages 10–19. ACM, 2019. doi:10.1145/3341163.3347735.

[24] V. Becker, S. Kalbermatter, S. Mayer, and G. Sörös. Tailored controls: Creating personalized tangible user interfaces from paper. In *Proceedings of the International Conference on Interactive Surfaces and Spaces*, ISS '19, New York, NY, USA,

2019. ACM. doi:10.1145/3343055.3359700.

[25] V. Becker and W. Kleiminger. Exploring zero-training algorithms for occupancy detection based on smart meter measurements. *Computer Science - Research and Development*, 33(1):25–36, 2017. doi:10.1007/s00450-017-0344-9.

[26] V. Becker, W. Kleiminger, V. C. Coroamă, and F. Mattern. Estimating the savings potential of occupancy-based heating strategies. *Energy Informatics*, 1(1):52, Oct 2018. doi:10.1186/s42162-018-0022-6.

[27] V. Becker, P. Oldrati, L. Barrios, and G. Sörös. TouchSense: Classifying and measuring the force of finger touches with an electromyography armband. In *Proceedings of the Augmented Human International Conference*, AH '18, pages 34:1–34:3, New York, NY, USA, 2018. ACM. Poster. doi:10.1145/3174910.3174947.

[28] V. Becker, P. Oldrati, L. Barrios, and G. Sörös. TouchSense: Classifying finger touches and measuring their force with an electromyography armband. In *Proceedings of the International Symposium on Wearable Computers*, ISWC '18, pages 1–8, New York, NY, USA, 2018. ACM. doi:10.1145/3267242.3267250.

[29] V. Becker, F. Rauchenstein, and G. Sörös. Connecting and controlling appliances through wearable augmented reality. *Augmented Human Research*, Aug 2019. doi:10.1007/s41133-019-0019-0.

[30] V. Becker, F. Rauchenstein, and G. Sörös. Investigating universal appliance control through wearable augmented reality. In *Proceedings of the Augmented Human International Conference*, AH '19, pages 41:1–41:9, New York, NY, USA, 2019. ACM. doi:10.1145/3311823.3311853.

[31] S. Benatti, F. Casamassima, B. Milosevic, E. Farella, P. Schönle, S. Fateh, T. Burger, Q. Huang, and L. Benini. A versatile embedded platform for EMG acquisition and gesture recognition. *IEEE Transactions on Biomedical Circuits and Systems*, 9(5):620–630, Oct 2015. doi:10.1109/TBCAS.2015.2476555.

[32] H. Benko, T. S. Saponas, D. Morris, and D. Tan. Enhancing input on and above the interactive surface with muscle sensing. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 93–100, New York, NY, USA, 2009. ACM. doi:10.1145/1731903.1731924.

[33] B. Berghmans, A. Faes, M. Kaminski, and K. Todi. Household survival: Immersive room-sized gaming using everyday objects as weapons. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA

'16, pages 168–171, New York, NY, USA, 2016. ACM. doi:10.1145/2851581.2890372.

[34] S.-J. Blakemore and J. Decety. From the perception of action to the understanding of intention. *Nature Reviews Neuroscience*, 2(8):561–567, 2001. doi:10.1038/35086023.

[35] F. Block, M. Haller, H. Gellersen, C. Gutwin, and M. Billinghurst. VoodooSketch: Extending interactive surfaces with adaptable interface palettes. In *Proceedings of the International Conference on Tangible and Embedded Interaction*, TEI '08, pages 55–58, New York, NY, USA, 2008. ACM. doi:10.1145/1347390.1347404.

[36] J. Bohn, V. Coroamă, M. Langheinrich, F. Mattern, and M. Rohs. Living in a world of smart everyday objects – social, economic, and ethical implications. *Human and Ecological Risk Assessment: An International Journal*, 10(5):763–785, 2004. doi:10.1080/10807030490513793.

[37] R. Boldu, H. Zhang, J. Cortés, S. Muthukumarana, and S. Nanayakkara. Insight: A systematic approach to create dynamic human-controller-interactions. In *Proceedings of the Augmented Human International Conference*, AH '17, pages 26:1–26:5, New York, NY, USA, 2017. ACM. doi:10.1145/3041164.3041195.

[38] A. Boyali, N. Hashimoto, and O. Matsumoto. Hand posture and gesture recognition using Myo armband and spectral collaborative representation based classification. In *IEEE Global Conference on Consumer Electronics*, GCCE '15, pages 200–201, October 2015. doi:10.1109/GCCE.2015.7398619.

[39] T. Brown. Design thinking. *Harvard business review*, 86:84–92, 141, 07 2008.

[40] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '16, pages 4960–4964, March 2016. doi:10.1109/ICASSP.2016.7472621.

[41] W. Chan, Y. Zhang, Q. Le, and N. Jaitly. Latent sequence decompositions. In *International Conference on Learning Representations*, ICLR '17, 2017. URL: https://arxiv.org/abs/1610.03035.

[42] X. Chen, X. Zhang, Z. Y. Zhao, J. H. Yang, V. Lantz, and K. Q. Wang. Hand gesture recognition research based on surface EMG sensors and 2D-accelerometers. In *Proceedings of the International Symposium on Wearable Computers*, ISWC '07, pages 11–14, Oct 2007. doi:10.1109/ISWC.2007.4373769.

[43] K. Cheng, R. Liang, B. Chen, R. Laing, and S. Kuo. iCon: Utilizing everyday objects as additional, auxiliary and instant tabletop controllers. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '10, pages 1155–1164, New York, NY, USA, 2010. ACM. doi:10.1145/1753326.1753499.

[44] J. Chorowski and N. Jaitly. Towards better decoding and language model integration in sequence to sequence models. In *Annual Conference of the International Speech Communication Association*, INTERSPEECH '17, 2017. URL: https://arxiv.org/abs/1612.02695.

[45] D. Clark. Network nirvana and the intelligent device. *IEEE Concurrency*, 7(2):16–19, Apr. 1999. doi:10.1109/4434.766980.

[46] C. Clarke, A. Bellino, A. Esteves, and H. Gellersen. Remote control by body movement in synchrony with orbiting widgets: An evaluation of TraceMatch. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3):45:1–45:22, Sep 2017. doi:10.1145/3130910.

[47] C. Clarke, A. Bellino, A. Esteves, E. Velloso, and H. Gellersen. TraceMatch: A computer vision technique for user input by tracing of animated controls. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, pages 298–303, New York, NY, USA, 2016. ACM. doi:10.1145/2971648.2971714.

[48] J. Clement. Percentage of mobile device website traffic worldwide from 1st quarter 2015 to 2nd quarter 2019. URL: https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/ [cited 23.03.2020].

[49] M. Coelho and J. Zigelbaum. Shape-changing interfaces. *Personal Ubiquitous Comput.*, 15(2):161–173, Feb. 2011. doi:10.1007/s00779-010-0311-y.

[50] A. Colley and J. Häkkilä. Exploring finger specific touch screen interaction for mobile phone user interfaces. In *Proceedings of the Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design*, OzCHI '14, pages 539–548, New York, NY, USA, 2014. ACM. doi:10.1145/2686612.2686699.

[51] C. Corsten, I. Avellino, M. Möllers, and J. Borchers. Instant user interfaces: Repurposing everyday objects as input devices. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 71–80, New York, NY, USA, 2013. ACM. doi:10.1145/2512349.2512799.

[52] B. R. Cowan, N. Pantidi, D. Coyle, K. Morrissey, P. Clarke, S. Al-Shehri, D. Earley, and N. Bandeira. "What can I help you with?": Infrequent users' experiences of intelligent personal assistants. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, pages 43:1–43:12. ACM, 2017. doi:10.1145/3098279.3098539.

[53] A. de Freitas, M. Nebeling, X. Chen, J. Yang, K. Ranithangam, A. Kirupa, and A. Dey. Snap-to-it: A user-inspired platform for opportunistic device interactions. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '16, pages 5909–5920, New York, NY, USA, 2016. ACM. doi:10.1145/2858036.2858177.

[54] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '13, pages 8599–8603, May 2013. doi:10.1109/ICASSP.2013.6639344.

[55] A. K. Dey, P. Ljungstrand, and A. Schmidt. Distributed and disappearing user interfaces in ubiquitous computing. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pages 487–488, New York, NY, USA, 2001. ACM. doi:10.1145/634067.634346.

[56] A. DiDomenico and M. A. Nussbaum. Estimation of forces exerted by the fingers using standardised surface electromyography from the forearm. *Ergonomics*, 51(6):858–871, 2008. PMID: 18484400. doi:10.1080/00140130801915980.

[57] C. Dimitrakakis and S. Bengio. Boosting HMMs with an application to speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, May 2004. doi:10.1109/ICASSP.2004.1327187.

[58] I. Donovan, J. Puchin, K. Okada, and X. Zhang. Simple space-domain features for low-resolution sEMG pattern recognition. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, EMBC '17, 2017. URL: https://bidal.sfsu.edu/~kazokada/research/okada_embc17_myoFeature.pdf.

[59] H. Drewes and A. Schmidt. Interacting with the computer using gaze gestures. In *Human-Computer Interaction – INTERACT Part II*, pages 475–488, Berlin, Heidelberg, 2007. Springer. URL: https://link.springer.com/chapter/10.1007/978-3-540-74800-7_43.

[60] Y. Du, W. Jin, W. Wei, Y. Hu, and W. Geng. Surface EMG-based inter-session gesture recognition enhanced by deep domain adaptation. *Sensors*, 17(3), 2017.

doi:10.3390/s17030458.

[61] Y. Du, Y. Wong, W. Jin, W. Wei, Y. Hu, M. Kankanhalli, and W. Geng. Semi-supervised learning for surface EMG-based gesture recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 1624–1630. AAAI Press, 2017. URL: http://dl.acm.org/citation.cfm?id=3172077.3172113.

[62] F. Echtler, M. Huber, and G. Klinker. Shadow tracking on multi-touch tables. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, pages 388–391, New York, NY, USA, 2008. ACM. doi:10.1145/1385569.1385640.

[63] P. Ehrlich and A. Ehrlich. *The Dominant Animal*. Island Press, 2008.

[64] K. Englehart and B. Hudgins. A robust, real-time control scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, 50(7):848–854, July 2003. doi:10.1109/TBME.2003.813539.

[65] A. Esteves, E. Velloso, A. Bulling, and H. Gellersen. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '15, pages 457–466, New York, NY, USA, 2015. ACM. doi:10.1145/2807442.2807499.

[66] L. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510 – 523, 1988. doi:10.1016/0013-4694(88)90149-6.

[67] G. W. Fitzmaurice, H. Ishii, and W. A. S. Buxton. Bricks: Laying the foundations for graspable user interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '95, pages 442–449, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. doi:10.1145/223904.223964.

[68] S. Follmer, D. Leithinger, A. Olwal, A. Hogge, and H. Ishii. inFORM: Dynamic physical affordances and constraints through shape and object actuation. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '13, pages 417–426, New York, NY, USA, 2013. ACM. doi:10.1145/2501988.2502032.

[69] K. Fujinami, M. Kosaka, and B. Indurkhya. Painting an apple with an apple: A tangible tabletop interface for painting with physical objects. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(4):162:1–162:22, Dec. 2018. doi:10.1145/3287040.

[70] M. Fukumoto and Y. Suenaga. "FingeRing": A full-time wearable interface. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '94, pages 81–82, New York, NY, USA, 1994. ACM. doi:10.1145/259963.260056.

[71] M. Funk, O. Korn, and A. Schmidt. An augmented workplace for enabling user-defined tangibles. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14, pages 1285–1290, New York, NY, USA, 2014. ACM. doi:10.1145/2559206.2581142.

[72] M. Gazzoni, N. Celadon, D. Mastrapasqua, M. Paleari, V. Margaria, and P. Ariano. Quantifying forearm muscle activity during wrist and finger movements by means of multi-channel electromyography. *PLOS ONE*, 9(10):1–11, 10 2014. doi:10.1371/journal.pone.0109943.

[73] M. Georgi, C. Amma, and T. Schultz. Recognizing hand and finger gestures with IMU based motion and EMG based muscle activity sensing. In *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing*, BIOSTEC '15, pages 99–108, 2015. doi:10.5220/0005276900990108.

[74] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, ICML'14, pages II–1764–II–1772. JMLR.org, 2014. URL: http://dl.acm.org/citation.cfm?id=3044805.3045089.

[75] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013. doi:10.1109/ICASSP.2013.6638947.

[76] A. Gupta and R. Balakrishnan. Dualkey: Miniature screen text entry via finger identification. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '16, pages 59–70, New York, NY, USA, 2016. ACM. doi:10.1145/2858036.2858052.

[77] S. Gupta, D. Morris, S. Patel, and D. Tan. Soundwave: Using the Doppler effect to sense gestures. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '12, pages 1911–1914, New York, NY, USA, 2012. ACM. doi:10.1145/2207676.2208331.

[78] T. Han, K. Hasan, K. Nakamura, R. Gomez, and P. Irani. Soundcraft: Enabling spatial interactions on smartwatches using hand generated acoustics. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '17, pages 579–591, New York, NY, USA, 2017. ACM. doi:10.1145/3126594.3126612.

[79] S. Henderson and S. Feiner. Opportunistic controls: Leveraging natural affordances as tangible user interfaces for augmented reality. In *Proceedings of the Symposium on Virtual Reality Software and Technology*, VRST '08, pages 211–218, New York, NY, USA, 2008. ACM. doi:10.1145/1450579.1450625.

[80] C. Herff, D. Heger, A. de Pesters, D. Telaar, P. Brunner, G. Schalk, and T. Schultz. Brain-to-text: decoding spoken phrases from phone representations in the brain. *Frontiers in Neuroscience*, 9:217, 2015. doi:10.3389/fnins.2015.00217.

[81] A. Hershman, J. Nazare, J. Qi, M. Saveski, D. Roy, and M. Resnick. Light it up: Using paper circuitry to enhance low-fidelity paper prototypes for children. In *Proceedings of the Conference on Interaction Design and Children*, IDC '18, pages 365–372, New York, NY, USA, 2018. ACM. doi:10.1145/3202185.3202758.

[82] V. Heun, J. Hobin, and P. Maes. Reality editor: Programming smarter objects. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '13 Adjunct, pages 307–310, New York, NY, USA, 2013. ACM. doi:10.1145/2494091.2494185.

[83] V. Heun, S. Kasahara, and P. Maes. Smarter objects: Using AR technology to program physical objects and their interactions. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 2939–2942, New York, NY, USA, 2013. ACM. doi:10.1145/2468356.2479579.

[84] G. Hickok and D. Poeppel. The cortical organization of speech processing. *Nat. Rev. Neurosci.*, 8:393–402, 2007. doi:10.1038/nrn2113.

[85] C. Holz and P. Baudisch. Fiberio: A touchscreen that senses fingerprints. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '13, pages 41–50, New York, NY, USA, 2013. ACM. doi:10.1145/2501988.2502021.

[86] K. Huo, Y. Cao, S. Yoon, Z. Xu, G. Chen, and K. Ramani. Scenariot: Spatially mapping smart things within augmented reality scenes. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '18, pages 219:1–219:13, New York, NY, USA, 2018. ACM. doi:10.1145/3173574.3173793.

[87] H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '97, pages 234–241, New York, NY, USA, 1997. ACM. doi:10.1145/258549.258715.

[88] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, April 1976. doi:10.1109/PROC.1976.10159.

[89] A. Karrenbauer and A. Oulasvirta. Improvements to keyboard optimization with integer programming. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '14, pages 621–626, New York, NY, USA, 2014. ACM. doi:10.1145/2642918.2647382.

[90] S. Kasahara, R. Niiyama, V. Heun, and H. Ishii. exTouch: Spatially-aware embodied manipulation of actuated objects mediated by augmented reality. In *Proceedings of the International Conference on Tangible, Embedded and Embodied Interaction*, TEI '13, pages 223–228, New York, NY, USA, 2013. ACM. doi:10.1145/2460625.2460661.

[91] Y. Kawahara, S. Hodges, B. S. Cook, C. Zhang, and G. D. Abowd. Instant inkjet circuits: Lab-based inkjet printing to support rapid prototyping of ubicomp devices. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 363–372, New York, NY, USA, 2013. ACM. doi:10.1145/2493432.2493486.

[92] J. Kaye, N. Matsakis, M. Gray, A. Wheeler, and M. Hawley. PC Dinners, Mr. Java and Counter Intelligence: Prototyping smart appliances for the kitchen. Technical report, MIT Media Laboratory, 2000. URL: http://alumni.media.mit.edu/~jofish/writing/information.appliances.pdf.

[93] A. Kelly, R. B. Shapiro, J. de Halleux, and T. Ball. ARcadia: A rapid prototyping platform for real-time tangible interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '18, pages 409:1–409:8, New York, NY, USA, 2018. ACM. doi:10.1145/3173574.3173983.

[94] F. Kerber, P. Lessel, and A. Krüger. Same-side hand interactions with arm-placed devices using EMG. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 1367–1372, New York, NY, USA, 2015. ACM. doi:10.1145/2702613.2732895.

[95] F. Kerber, M. Puhl, and A. Krüger. User-independent real-time hand gesture recognition based on surface electromyography. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, pages 36:1–36:7, New York, NY, USA, 2017. ACM. doi:10.1145/3098279.3098553.

[96] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, J. Chen, I. Oikonomidis, and P. Olivier. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In

Proceedings of the Symposium on User Interface Software and Technology, UIST '12, pages 167–176, New York, NY, USA, 2012. ACM. doi:10.1145/2380116.2380139.

[97] D. Kim, S. Izadi, J. Dostal, C. Rhemann, C. Keskin, C. Zach, J. Shotton, T. Large, S. Bathiche, M. Niessner, D. A. Butler, S. Fanello, and V. Pradeep. Retrodepth: 3D silhouette sensing for high-precision input on and above physical surfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '14, pages 1377–1386, New York, NY, USA, 2014. ACM. doi:10.1145/2556288.2557336.

[98] H. Kim. Fostering design process of shape-changing interfaces. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '18 Adjunct, pages 224–227, New York, NY, USA, 2018. ACM. doi:10.1145/3266037.3266131.

[99] H. Kim, C. Coutrix, and A. Roudaut. Morphees+: Studying everyday reconfigurable objects for the design and taxonomy of reconfigurable UIs. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '18, pages 619:1–619:14, New York, NY, USA, 2018. ACM. doi:10.1145/3173574.3174193.

[100] J. Kim, N. D. Thang, and T. Kim. 3-D hand motion tracking and gesture recognition using a data glove. In *IEEE International Symposium on Industrial Electronics*, pages 1013–1018, July 2009. doi:10.1109/ISIE.2009.5221998.

[101] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, ICLR '15, 2015. URL: https://arxiv.org/abs/1412.6980.

[102] B. Kriesten, C. Mertes, R. Tünnermann, and T. Hermann. Unobtrusively controlling and linking information and services in smart environments. In *Proceedings of the Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pages 276–285, New York, NY, USA, 2010. ACM. doi:10.1145/1868914.1868948.

[103] T. Kubitza. Apps for environments: Demonstrating pluggable apps for multi-device IoT-setups. In *Proceedings of the International Conference on the Internet of Things*, IoT'16, pages 185–186, New York, NY, USA, 2016. ACM. doi:10.1145/2991561.2998473.

[104] T. Kubitza and A. Schmidt. Towards a toolkit for the rapid creation of smart environments. In *End-User Development*, pages 230–235, Cham, 2015. Springer. URL: https://link.springer.com/chapter/10.1007/978-3-319-18425-8_21.

[105] K. Kunze, M. Iwamura, K. Kise, S. Uchida, and S. Omachi. Activity recognition for the mind: Toward a cognitive "quantified self". *Computer*, 46(10):105–108, October 2013. doi:10.1109/MC.2013.339.

[106] N. Lane, P. Georgiev, and L. Qendro. DeepEar: Robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 283–294, New York, NY, USA, 2015. ACM. doi:10.1145/2750858.2804262.

[107] M. Langheinrich, F. Mattern, K. Römer, and H. Vogt. First steps towards an event-based infrastructure for smart things. In *Ubiquitous Computing Workshop (PACT 2000)*, Philadelphia, PA, Oct 2000. URL: http://www.vs.inf.ethz.ch/publ/papers/firststeps.pdf.

[108] G. Laput, K. Ahuja, M. Goel, and C. Harrison. Ubicoustics: Plug-and-play acoustic activity recognition. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '18, pages 213–224, New York, NY, USA, 2018. ACM. doi:10.1145/3242587.3242609.

[109] G. Laput, R. Xiao, and C. Harrison. Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '16, pages 321–333, New York, NY, USA, 2016. ACM. doi:10.1145/2984511.2984582.

[110] B. Laugwitz, T. Held, and M. Schrepp. Construction and evaluation of a user experience questionnaire. In *HCI and Usability for Education and Work*, USAB '08, pages 63–76, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-89350-9_6.

[111] D. Ledo, S. Greenberg, N. Marquardt, and S. Boring. Proxemic-aware controls: Designing remote controls for ubiquitous computing ecologies. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 187–198, New York, NY, USA, 2015. ACM. doi:10.1145/2785830.2785871.

[112] D. Leithinger, S. Follmer, A. Olwal, S. Luescher, A. Hogge, J. Lee, and H. Ishii. Sublimate: State-changing virtual and physical rendering to augment interaction with shape displays. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '13, pages 1441–1450, New York, NY, USA, 2013. ACM. doi:10.1145/2470654.2466191.

[113] H. Li, W. Yang, J. Wang, Y. Xu, and L. Huang. WiFinger: Talk to your smart devices

with finger-grained gesture. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, pages 250–261, New York, NY, USA, 2016. ACM. doi:10.1145/2971648.2971738.

[114] Y. Li, X. Chen, J. Tian, X. Zhang, K. Wang, and J. Yang. Automatic recognition of sign language subwords based on portable accelerometer and EMG sensors. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, ICMI-MLMI '10, pages 17:1–17:7, New York, NY, USA, 2010. ACM. doi:10.1145/1891903.1891926.

[115] S. Lin, H. Cheng, W. Li, Z. Huang, P. Hui, and C. Peylo. Ubii: Physical world interaction through augmented reality. *IEEE Trans. on Mobile Computing*, 16(3):872–885, March 2017. doi:10.1109/TMC.2016.2567378.

[116] S. Liu, Z. Zhou, J. Du, L. Shangguan, J. Han, and X. Wang. UbiEar: Bringing location-independent sound awareness to the hard-of-hearing people with smartphones. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(2):17:1–17:21, June 2017. doi:10.1145/3090082.

[117] B. T. Lowerre. *The Harpy Speech Recognition System.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1976. AAI7619331.

[118] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[119] S. Malik, A. Ranjan, and R. Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '05, pages 43–52, New York, NY, USA, 2005. ACM. doi:10.1145/1095034.1095042.

[120] N. Marquardt, J. Kiemer, D. Ledo, S. Boring, and S. Greenberg. Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces*, ITS '11, pages 21–30, New York, NY, USA, 2011. ACM. doi:10.1145/2076354.2076358.

[121] D. Masson, A. Goguey, S. Malacria, and G. Casiez. WhichFingers: Identifying fingers on touch surfaces and keyboards using vibration sensors. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '17, pages 41–48, New York, NY, USA, 2017. ACM. doi:10.1145/3126594.3126619.

[122] F. Mattern. From smart devices to smart everyday objects. In *Proceedings of the Smart Objects Conference*, SOC '03, pages 15–16, 2003. URL: http://www.vs.inf.ethz.ch/publ/papers/Generic_106.pdf.

[123] S. Mayer, Y. Hassan, and G. Sörös. A magic lens for revealing device interactions in smart environments. In *SIGGRAPH Asia Mobile Graphics and Interactive Applications*, SA '14, pages 9:1–9:6, New York, NY, USA, 2014. ACM. doi:10.1145/2669062.2669077.

[124] S. Mayer and G. Sörös. User interface beaming - seamless interaction with smart things using personal wearable computers. In *Proceedings of the International Conference on Wearable and Implantable Body Sensor Networks, Workshop on Glass & Eyewear Computers*, BSN '14, pages 46–49, Jun 2014. doi:10.1109/BSN.Workshops.2014.17.

[125] S. Mayer, A. Tschofen, A. Dey, and F. Mattern. User interfaces for smart things – a generative approach with semantic interaction descriptions. *ACM Trans. Comput.-Hum. Interact.*, 21(2):12:1–12:25, Feb 2014. doi:10.1145/2584670.

[126] D. J. McFarland, L. A. Miner, T. M. Vaughan, and J. R. Wolpaw. Mu and beta rhythm topographies during motor imagery and actual movements. *Brain Topography*, 12(3):177–186, Mar 2000. doi:10.1023/A:1023437823106.

[127] J. McIntosh, C. McNeill, M. Fraser, F. Kerber, M. Löchtefeld, and A. Krüger. EMPress: Practical hand gesture classification with wrist-mounted EMG and pressure sensing. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '16, pages 2332–2342, New York, NY, USA, 2016. ACM. doi:10.1145/2858036.2858093.

[128] E. Mollenbach, J. P. Hansen, M. Lillholm, and A. G. Gale. Single stroke gaze gestures. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 4555–4560, New York, NY, USA, 2009. ACM. doi:10.1145/1520340.1520699.

[129] S. Murugappan, Vinayak, N. Elmqvist, and K. Ramani. Extended multitouch: Recovering touch posture and differentiating users using a depth camera. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '12, pages 487–496, New York, NY, USA, 2012. ACM. doi:10.1145/2380116.2380177.

[130] G. R. Naik, D. K. Kumar, V. P. Singh, and M. Palaniswami. Hand gestures for HCI using ICA of EMG. In *Proceedings of the HCSNet Workshop on Use of Vision in Human-computer Interaction - Volume 56*, VisHCI '06, pages 67–72,

Darlinghurst, Australia, Australia, 2006. Australian Computer Society Inc. URL: http://dl.acm.org/citation.cfm?id=1273385.1273397.

[131] D. A. Norman. *The Design of Everyday Things*. Basic Books Inc., New York, NY, USA, 2002.

[132] M. Oberweger and V. Lepetit. DeepPrior++: Improving fast and accurate 3D hand pose estimation. In *International Conference on Computer Vision Workshops*, 2017.

[133] S. Odashima, T. Kanaoka, K. Miura, K. Okabayashi, and N. Sawasaki. Human activeness recognition by variety of rare sounds. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, pages 181–184, New York, NY, USA, 2016. ACM. doi:10.1145/2968219.2971400.

[134] S. Olberding, N.-W. Gong, J. Tiab, J. A. Paradiso, and J. Steimle. A cuttable multi-touch sensor. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '13, pages 245–254, New York, NY, USA, 2013. ACM. doi:10.1145/2501988.2502048.

[135] H. Park, K. Koh, Y. Choi, H. Chae, J. Hwang, and J. Seo. Defining rules among devices in smart environment using an augmented reality headset. In *Proceedings of the International Conference on IoT in Urban Space*, Urb-IoT '16, pages 18–21, New York, NY, USA, 2016. ACM. doi:10.1145/2962735.2962746.

[136] V. Perumal and D. Wigdor. Printem: Instant printed circuit boards with standard office printers & inks. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '15, pages 243–251, New York, NY, USA, 2015. ACM. doi:10.1145/2807442.2807511.

[137] A. Phinyomark, C. Limsakul, and P. Phukpattaranont. A novel feature extraction for robust EMG pattern recognition. *CoRR*, abs/0912.3973, 2009. URL: http://arxiv.org/abs/0912.3973.

[138] H. Pohl and M. Rohs. Around-device devices: My coffee mug is a volume dial. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '14, pages 81–90, New York, NY, USA, 2014. ACM. doi:10.1145/2628363.2628401.

[139] G. Privat. A system-architecture viewpoint on smart networked devices. *Microelectronic Engineering*, 54(1):193–197, 2000. doi:10.1016/S0167-9317(00)80070-2.

[140] Q. Pu, S. Gupta, S. Gollakota, and S. Patel. Whole-home gesture recognition using wireless signals. In *Proceedings of the Annual International Conference on Mobile Computing and Networking*, MobiCom '13, pages 27–38, New York, NY, USA, 2013. ACM. doi:10.1145/2500423.2500436.

[141] J. Qi and L. Buechley. Sketching in circuits: Designing and building electronics on paper. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '14, pages 1713–1722, New York, NY, USA, 2014. ACM. doi:10.1145/2556288.2557391.

[142] J. Qi, A. Demir, and J. A. Paradiso. Code collage: Tangible programming on paper with circuit stickers. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 1970–1977, New York, NY, USA, 2017. ACM. doi:10.1145/3027063.3053084.

[143] R. Ramakers, K. Todi, and K. Luyten. Paperpulse: An integrated approach for embedding electronics in paper designs. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '15, pages 2457–2466, New York, NY, USA, 2015. ACM. doi:10.1145/2702123.2702487.

[144] D. Rodriguez, A. Piryatinska, and X. Zhang. A neural decision forest scheme with application to EMG gesture classification. In *Science and Information Computing Conference*, pages 243–252, July 2016. doi:10.1109/SAI.2016.7555990.

[145] O. R. Roup. Hive: a software infrastructure for things that think. Master's thesis, Massachusetts Institute of Technology, 1999. URL: http://hdl.handle.net/1721.1/9155.

[146] H. Sak, A. Senior, K. Rao, F. Beaufays, and J. Schalkwyk. Google voice search: faster and more accurate. URL: https://ai.googleblog.com/2015/09/google-voice-search-faster-and-more.html [cited 23.03.2020].

[147] A. A. Samadani and D. Kulic. Hand gesture recognition based on surface electromyography. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, EMBC '14, pages 4196–4199, Aug 2014. doi:10.1109/EMBC.2014.6944549.

[148] T. S. Saponas, D. S. Tan, D. Morris, and R. Balakrishnan. Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '08, pages 515–524, New York, NY, USA, 2008. ACM. doi:10.1145/1357054.1357138.

[149] T. S. Saponas, D. S. Tan, D. Morris, J. Turner, and J. A. Landay. Making muscle-computer interfaces more practical. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '10, pages 851–854, New York, NY, USA, 2010. ACM. doi:10.1145/1753326.1753451.

[150] J. Sargent, M. Clarke, K. Price, T. Griffiths, and J. Swettenham. Use of eye-pointing by children with cerebral palsy: what are we looking at? *International Journal of Language & Communication Disorders*, 48(5):477–485, 2013. doi:10.1111/1460-6984.12026.

[151] E. Scheme and K. Englehart. On the robustness of EMG features for pattern recognition based myoelectric control; a multi-dataset comparison. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, EMBC '14, pages 650–653, Aug 2014. doi:10.1109/EMBC.2014.6943675.

[152] D. Schmidt, D. Molyneaux, and X. Cao. PICOntrol: Using a handheld projector for direct control of physical devices through visible light. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '12, pages 379–388, New York, NY, USA, 2012. ACM. doi:10.1145/2380116.2380166.

[153] M. Schrapel, M.-L. Stadler, and M. Rohs. Pentelligence: Combining pen tip motion and writing sounds for handwritten digit recognition. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '18, pages 131:1–131:11, New York, NY, USA, 2018. ACM. doi:10.1145/3173574.3173705.

[154] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL: http://arxiv.org/abs/1409.1556, arXiv:1409.1556.

[155] J. Song, F. Pece, G. Sörös, M. Koelle, and O. Hilliges. Joint estimation of 3D hand position and gestures from monocular video for mobile interaction. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '15, pages 3657–3660, New York, NY, USA, 2015. ACM. doi:10.1145/2702123.2702601.

[156] J. Song, G. Sörös, F. Pece, S. Fanello, S. Izadi, C. Keskin, and O. Hilliges. In-air gestures around unmodified mobile devices. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '14, pages 319–329, New York, NY, USA, 2014. ACM. doi:10.1145/2642918.2647373.

[157] J. Sorgalla, J. Fleck, and S. Sachweh. ARGI: Augmented reality for gesture-based interaction in variable smart environments. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graph-*

ics Theory and Applications, VISIGRAPP '18, pages 102–107, 2018. doi:10.5220/0006621301020107.

[158] O. Sorkine-Hornung and M. Rabinovich. Least-squares rigid motion using SVD. Technical report, ETH Zurich, 2017. URL: https://igl.ethz.ch/projects/ARAP/svd_rot.pdf.

[159] M. Spindler, V. Cheung, and R. Dachselt. Dynamic tangible user interface palettes. In *Human-Computer Interaction – INTERACT 2013*, pages 159–176, Berlin, Heidelberg, 2013. Springer. URL: https://link.springer.com/chapter/10.1007/978-3-642-40498-6_12.

[160] D. J. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, Jan 1994. doi:10.1109/38.250916.

[161] K. Sun, Y. Wang, C. Yu, Y. Yan, H. Wen, and Y. Shi. Float: One-handed and touch-free target selection on smartwatches. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '17, pages 692–704, New York, NY, USA, 2017. ACM. doi:10.1145/3025453.3026027.

[162] B. Tag, A. W. Vargo, A. Gupta, G. Chernyshov, K. Kunze, and T. Dingler. Continuous alertness assessments: Using EOG glasses to unobtrusively monitor fatigue levels in-the-wild. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '19, pages 464:1–464:12, New York, NY, USA, 2019. ACM. doi:10.1145/3290605.3300694.

[163] K. N. Tarchanidis and J. N. Lygouras. Data glove with a force sensor. *IEEE Transactions on Instrumentation and Measurement*, 52(3):984–989, June 2003. doi:10.1109/TIM.2003.809484.

[164] D. Tennenhouse. Proactive computing. *Commun. ACM*, 43(5):43–50, May 2000. doi:10.1145/332833.332837.

[165] M. Tomasello, M. Carpenter, J. Call, T. Behne, and H. Moll. Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and Brain Sciences*, 28(5):675–691, 2005. doi:10.1017/S0140525X05000129.

[166] K. Vega and H. Fuks. Beauty tech nails: Interactive technology at your fingertips. In *Proceedings of the International Conference on Tangible, Embedded and Embodied Interaction*, TEI '14, pages 61–64, New York, NY, USA, 2013. ACM. doi:10.1145/2540930.2540961.

[167] E. Velloso, M. Wirth, C. Weichel, A. Esteves, and H. Gellersen. AmbiGaze: Direct

control of ambient devices by gaze. In *Proceedings of the Conference on Designing Interactive Systems*, DIS '16, pages 812–817, New York, NY, USA, 2016. ACM. doi:10.1145/2901790.2901867.

[168] D. Ververidis, S. Karavarsamis, S. Nikolopoulos, and I. Kompatsiaris. Pottery gestures style comparison by exploiting Myo sensor and forearm anatomy. In *Proceedings of the International Symposium on Movement and Computing*, MOCO '16, pages 3:1–3:8, New York, NY, USA, 2016. ACM. doi:10.1145/2948910.2948924.

[169] M. Vidal, A. Bulling, and H. Gellersen. Pursuits: Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 439–448, New York, NY, USA, 2013. ACM. doi:10.1145/2493432.2493477.

[170] N. Villar, D. Cletheroe, G. Saul, C. Holz, T. Regan, O. Salandin, M. Sra, H.-S. Yeo, W. Field, and H. Zhang. Project Zanzibar: A portable and flexible tangible interaction platform. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '18, pages 515:1–515:13, New York, NY, USA, 2018. ACM. doi:10.1145/3173574.3174089.

[171] S. Voelker, K. Nakajima, C. Thoresen, Y. Itoh, K. I. Overgaard, and J. Borchers. PUCs: Detecting transparent, passive untouched capacitive widgets on unmodified multi-touch displays. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 101–104, New York, NY, USA, 2013. ACM. doi:10.1145/2512349.2512791.

[172] M. Vrigkas, C. Nikou, and I. Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and Artificial Intelligence*, 2, 11 2015. doi:10.3389/frobt.2015.00028.

[173] P. Wagner, Z. Malisz, and S. Kopp. Gesture and speech in interaction: An overview. *Speech Communication*, 57:209–232, 2014. doi:10.1016/j.specom.2013.09.008.

[174] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, March 1989. doi:10.1109/29.21701.

[175] M. Wand, C. Schulte, M. Janke, and T. Schultz. Array-based electromyographic silent speech interface. In *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing*, 2013. BIOSIGNALS

2013. URL: https://www.csl.uni-bremen.de/cms/images/documents/publications/BS13_WandSchulteJankeSchultz_ArrayBasedEMGSSI.pdf.

[176] M. Wand. and T. Schultz. Session-independent EMG-based speech recognition. In *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing - Volume 1: BIOSIGNALS*, BIOSTEC 2011, pages 295–300. INSTICC, SciTePress, 2011. doi:10.5220/0003169702950300.

[177] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges. Interacting with Soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum. In *Proceedings of the Symposium on User Interface Software and Technology*, UIST '16, pages 851–860, New York, NY, USA, 2016. ACM. doi:10.1145/2984511.2984565.

[178] J. A. Ward, P. Lukowicz, and G. Tröster. Gesture spotting using wrist worn microphone and 3-axis accelerometer. In *Proceedings of the Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, sOc-EUSAI '05, pages 99–104, New York, NY, USA, 2005. ACM. doi:10.1145/1107548.1107578.

[179] S. Warren, R. L. Craft, and J. T. Bosma. Designing smart health care technology into the home of the future. Technical report, Sandia National Labs, Albuquerque, NM, US, 1999.

[180] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, Sept. 1991. doi:10.1038/scientificamerican0991-94.

[181] M. Weiser and J. S. Brown. *The Coming Age of Calm Technology*, pages 75–85. Springer, New York, NY, 1997. doi:10.1007/978-1-4612-0685-9_6.

[182] M. Weiss, F. Mattern, T. Graml, T. Staake, and E. Fleisch. Handy feedback: Connecting smart meters with mobile phones. In *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*, MUM '09, pages 15:1–15:4, New York, NY, USA, 2009. ACM. doi:10.1145/1658550.1658565.

[183] M. Weiss, J. Wagner, Y. Jansen, R. Jennings, R. Khoshabeh, J. D. Hollan, and J. Borchers. Slap widgets: Bridging the gap between virtual and physical controls on tabletops. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '09, pages 481–490, New York, NY, USA, 2009. ACM. doi:10.1145/1518701.1518779.

[184] P. Wellner, W. Mackay, and R. Gold. Back to the real world. *Commun. ACM*, 36(7):24–26, July 1993. doi:10.1145/159544.159555.

[185] H. Wen, J. Ramos Rojas, and A. K. Dey. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '16, pages 3847–3851, New York, NY, USA, 2016. ACM. doi:10.1145/2858036.2858466.

[186] M. L. West. *Ancient Greek Music*. Clarendon paperbacks. Clarendon Press, 1994.

[187] C. Winkler, M. Löchtefeld, D. Dobbelstein, A. Krüger, and E. Rukzio. SurfacePhone: A mobile projection device for single- and multiuser everywhere tabletop interaction. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '14, pages 3513–3522, New York, NY, USA, 2014. ACM. doi:10.1145/2556288.2557075.

[188] R. Xiao, S. Hudson, and C. Harrison. Direct: Making touch tracking on ordinary surfaces practical with hybrid depth-infrared sensing. In *Proceedings of the International Conference on Interactive Surfaces and Spaces*, ISS '16, pages 85–94, New York, NY, USA, 2016. ACM. doi:10.1145/2992154.2992173.

[189] R. Xiao, G. Laput, Y. Zhang, and C. Harrison. Deus em machina: On-touch contextual functionality for smart IoT appliances. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '17, pages 4000–4008, New York, NY, USA, 2017. ACM. doi:10.1145/3025453.3025828.

[190] K. Yatani and K. N. Truong. Bodyscope: A wearable acoustic sensor for activity recognition. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '12, pages 341–350, New York, NY, USA, 2012. ACM. doi:10.1145/2370216.2370269.

[191] H.-S. Yeo, R. Minami, K. Rodriguez, G. Shaker, and A. Quigley. Exploring tangible interactions with radar sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(4):200:1–200:25, Dec. 2018. doi:10.1145/3287078.

[192] T. Yu, H. Jin, and K. Nahrstedt. WritingHacker: Audio based eavesdropping of handwriting via mobile devices. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, pages 463–473, New York, NY, USA, 2016. ACM. doi:10.1145/2971648.2971681.

[193] B. Zhang, Y. Chen, C. Tuna, A. Dave, Y. Li, E. Lee, and B. Hartmann. HOBS: Head orientation-based selection in physical spaces. In *Proceedings of the Symposium on Spatial User Interaction*, SUI '14, pages 17–25, New York, NY, USA, 2014. ACM. doi:10.1145/2659766.2659773.

[194] C. Zhang, A. Bedri, G. Reyes, B. Bercik, O. T. Inan, T. E. Starner, and G. D.

Abowd. TapSkin: Recognizing on-skin input for smartwatches. In *Proceedings of the International Conference on Interactive Surfaces and Spaces*, ISS '16, pages 13–22, New York, NY, USA, 2016. ACM. doi:10.1145/2992154.2992187.

[195] C. Zhang, A. Waghmare, P. Kundra, Y. Pu, S. Gilliland, T. Ploetz, T. E. Starner, O. T. Inan, and G. D. Abowd. FingerSound: Recognizing unistroke thumb gestures using a ring. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3):120:1–120:19, Sept. 2017. doi:10.1145/3130985.

[196] C. Zhang, Q. Xue, A. Waghmare, R. Meng, S. Jain, Y. Han, X. Li, K. Cunefare, T. Ploetz, T. Starner, O. Inan, and G. D. Abowd. FingerPing: Recognizing fine-grained hand poses using active acoustic on-body sensing. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '18, pages 437:1–437:10, New York, NY, USA, 2018. ACM. doi:10.1145/3173574.3174011.

[197] C. Zhang, J. Yang, C. Southern, T. E. Starner, and G. D. Abowd. WatchOut: Extending interactions on a smartwatch with inertial sensing. In *Proceedings of the International Symposium on Wearable Computers*, ISWC '16, pages 136–143, New York, NY, USA, 2016. ACM. doi:10.1145/2971763.2971775.

[198] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang. A framework for hand gesture recognition based on accelerometer and EMG sensors. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(6):1064–1076, Nov 2011. doi:10.1109/TSMCA.2011.2116004.

[199] X. Zhang, X. Chen, W.-H. Wang, J.-H. Yang, V. Lantz, and K.-Q. Wang. Hand gesture recognition and virtual game control based on 3D accelerometer and EMG sensors. In *Proceedings of the International Conference on Intelligent User Interfaces*, IUI '09, pages 401–406, New York, NY, USA, 2009. ACM. doi:10.1145/1502650.1502708.

[200] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Appearance-based gaze estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 4511–4520, June 2015. doi:10.1109/CVPR.2015.7299081.

[201] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. It's written all over your face: Full-face appearance-based gaze estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPR '17, pages 51–60, July 2017. URL: http://openaccess.thecvf.com/content_cvpr_2017_workshops/w41/papers/Bulling_Its_Written_All_CVPR_2017_paper.pdf.

[202] Y. Zhang and C. Harrison. Pulp nonfiction: Low-cost touch tracking for paper. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '18, pages 117:1–117:11, New York, NY, USA, 2018. ACM. doi:10.1145/3173574.3173691.

[203] J. Zheng and D. Vogel. Finger-aware shortcuts. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '16, pages 4274–4285, New York, NY, USA, 2016. ACM. doi:10.1145/2858036.2858355.

[204] F. Zhengren, G. Chernyshov, D. Zheng, and K. Kunze. Cognitive load assessment from facial temperature using smart eyewear. In *Adjunct Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the International Symposium on Wearable Computers*, UbiComp / ISWC '19 Adjunct, pages 657–660, New York, NY, USA, 2019. ACM. doi:10.1145/3341162.3348391.

[205] O. Zuckerman and A. Gal-Oz. To TUI or not to TUI: Evaluating performance and preference in tangible vs. graphical user interfaces. *International Journal of Human-Computer Studies*, 71(7):803–820, 2013. doi:10.1016/j.ijhcs.2013.04.003.

Web Resources

[206] Amazon. Alexa. URL: https://developer.amazon.com/en/alexa [cited 23.03.2020].

[207] P. N. Amsen. Noise library. URL: https://github.com/paramsen/noise [cited 23.03.2020].

[208] Apple. Siri. URL: https://www.apple.com/siri/ [cited 23.03.2020].

[209] V. Becker. Gestear repository. URL: https://github.com/vincentbecker/GestEar [cited 23.03.2020].

[210] V. Becker. GestEar video. URL: https://youtu.be/cfT4eOho6v4 [cited 23.03.2020].

[211] V. Becker. Interaction in augmented reality demo video. URL: https://youtu.be/oS52JkAO5n0 [cited 23.03.2020].

[212] V. Becker. Tailored Controls repository. URL: https://github.com/vincentbecker/TailoredControls [cited 23.03.2020].

[213] V. Becker. Tailored Controls video. URL: https://youtu.be/p_wS6BhTpQQ [cited 23.03.2020].

[214] V. Becker. TouchSense repository. URL: https://github.com/vincentbecker/TouchSense [cited 23.03.2020].

[215] V. Becker. TouchSense video. URL: https://youtu.be/BHB1jPdCdtg [cited 23.03.2020].

[216] Brack. Tobii 4C eye tracker. URL: https://www.brack.ch/tobii-eye-tracker-4c-491871?page=4 [cited 23.03.2020].

[217] Chibitronics Inc. Chibitronics homepage. URL: https://chibitronics.com [cited 23.03.2020].

[218] Communications & Multimedia Laboratory, National Taiwan University. iCon: Utilizing everyday objects as additional, auxiliary and instant tabletop controllers. URL: http://www.cmlab.csie.ntu.edu.tw/~howieliang/icon.html [cited 23.03.2020].

[219] CTRL-Labs. CTRL-kit. URL: https://www.ctrl-labs.com/ctrl-kit/ [cited 23.03.2020].

[220] Facebook IQ. The multidevice movement: Teens in France and Germany, 2016. URL: https://www.facebook.com/iq/articles/the-multidevice-movement-teens-in-france-and-germany?ref=wpinsights_rd [cited 23.03.2020].

[221] M. Gnauk. Free to use sounds. URL: https://www.freetousesounds.com [cited 23.03.2020].

[222] Google. Context hub runtime environment (CHRE). URL: https://android.googlesource.com/platform/system/chre/ [cited 23.03.2020].

[223] Google. Google assistant. URL: https://assistant.google.com/ [cited 23.03.2020].

[224] Google. Google home. URL: https://store.google.com/?srp=/product/google_home [cited 23.03.2020].

[225] A. Hinderks, M. Schrepp, and J. Thomaschewski. User experience questionnaire homepage. URL: http://www.ueq-online.org [cited 23.03.2020].

[226] IBM. IBM speech recognition. URL: https://www.ibm.com/topics/speech-recognition [cited 23.03.2020].

[227] IBM Emerging Technology Services. Node-RED. URL: https://nodered.org [cited 23.03.2020].

[228] Jins Meme. Jins Meme glasses. URL: https://jins-meme.com [cited 23.03.2020].

[229] M. Kellermann. Music player daemon homepage. URL: https://www.musicpd.org [cited 23.03.2020].

[230] Leap Motion. Leap motion. URL: https://www.leapmotion.com/ [cited 23.03.2020].

[231] LIFX. LIFX homepage. URL: https://www.lifx.com/ [cited 23.03.2020].

[232] Microsoft. Cortana. URL: https://www.microsoft.com/en-us/cortana [cited 23.03.2020].

[233] Microsoft. Microsoft Kinect. URL: https://developer.microsoft.com/en-us/windows/kinect [cited 23.03.2020].

[234] North. Focals. URL: https://www.bynorth.com/focals [cited 23.03.2020].

[235] PTC Inc. Vuforia homepage. URL: https://www.ptc.com/en/products/augmented-reality/vuforia [cited 23.03.2020].

[236] Pupil Labs. Pupil Labs Core. URL: https://pupil-labs.com/products/ [cited 23.03.2020].

[237] Qualcomm Technologies Inc. Qualcomm neural processing SDK for AI. URL: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk [cited 23.03.2020].

[238] Robotshop. Myo armband. URL: https://www.robotshop.com/de/de/myo-gestensteuerungsarmband-schwarz.html [cited 23.03.2020].

[239] J. Sissel. xdotool homepage. URL: https://www.semicomplete.com/projects/xdotool [cited 23.03.2020].

[240] Smithsonian National Museum of American History. Apple II Personal Computer. URL: https://www.si.edu/object/nmah_334638 [cited 23.03.2020].

[241] Synaptics. Force pad. URL: https://www.synaptics.com/products/touchpad-family/forcepad [cited 23.03.2020].

[242] TensorFlow. TensorFlow platform. URL: https://www.tensorflow.org/ [cited 23.03.2020].

[243] Thalmic Labs. Myo armband. URL: https://developerblog.myo.com/author/thalmic-labs/ [cited 23.03.2020].

[244] The Dynamic Medium Group. Dynamicland homepage. URL: https://dynamicland.org [cited 23.03.2020].

[245] Universität Stuttgart, Fachbereich Informatik. Die erste Computermaus - RKS 100. URL: https://www.f05.uni-stuttgart.de/informatik/fachbereich/computermuseum/aktuelles/Die-erste-Computermaus/ [cited 23.03.2020].

[246] B. Untiedt. Historische Mensch-Maschine-Schnittstelle einer Dampflokomotive. URL: https://de.wikipedia.org/wiki/Benutzerschnittstelle#/media/

Datei:P8-f%C3%BChrerstand2.jpg [cited 23.03.2020].

[247] Wikipedia. Force touch. URL: https://en.wikipedia.org/wiki/Force_Touch [cited 23.03.2020].

[248] Wikipedia. The Clapper. URL: https://en.wikipedia.org/wiki/The_Clapper [cited 23.03.2020].

[249] Wikipedia user SRI International. SRI computer mouse. URL: https://commons.wikimedia.org/w/index.php?curid=17294412 [cited 23.03.2020].

[250] Wikipedia user TobiToaster. Der Führerstand des ICE 3. URL: https://commons.wikimedia.org/wiki/File:Ice3_leitstand.jpg [cited 23.03.2020].

Short Curriculum Vitae

Vincent Georg Ernest Becker

Personal Data

Date of Birth	17th December 1991
Place of Birth	Stuttgart, Germany
Citizenship	German and British

Education

2016 – 2020	**Dr. sc. ETH Zurich** Department of Computer Science, ETH Zurich, Switzerland Thesis: Interacting with Smart Devices – Advancements in Gesture Recognition and Augmented Reality
2014 – 2016	**M.Sc.** in Informatics *(with distinction)* Karlsruhe Institute of Technology (KIT), Germany Thesis (at INESC Porto, Portugal): Concept Change Detection in Correlated Subspaces in Data Streams
2010 – 2014	**B.Sc.** in Informatics *(with distinction)* Karlsruhe Institute of Technology (KIT), Germany Thesis (at Carnegie Mellon University, Pittsburgh, US): Adaptation of a Cognitive Decision-Making Model through the Application of Workload Recognition
2002 – 2010	Friedrich-Schiller-Gymnasium Ludwigsburg, Germany Graduated with **Abitur** (general qualification for university entrance)

Professional Experience

2016 – 2020	Research and teaching assistant, ETH Zurich, Switzerland
2019	Software Engineering Intern, Google, London, UK
2012 – 2015	Co-Founder, Becker&Brenner&Liang GbR, Karlsruhe, Germany
2015	Working student, FARO Europe, Stuttgart, Germany
2014	Internship, FARO Europe, Stuttgart, Germany
2013	Research assistant, Cognitive Systems Labs, KIT, Germany
2012 – 2013	Teaching assistant, KIT, Germany
2010	Bundeswehr, Luftwaffenausbildungsregiment Mengen/Hohentengen, Germany (military service)

Publications

2020 [12] Mihai Bâce, **Vincent Becker**, Chenyang Wang, Andreas Bulling. Combining Gaze Estimation and Optical Flow for Pursuits Interaction. *ETRA '20*. The first three authors contributed equally.

2019 [24] **Vincent Becker**, Sandro Kalbermatter, Simon Mayer, Gábor Sörös. Tailored Controls: Creating Personalized Tangible User Interfaces from Paper. *ISS '19*.

2019 [23] **Vincent Becker**, Linus Fessler, Gábor Sörös. GestEar: Combining Audio and Motion Sensing for Gesture Recognition on Smartwatches. *ISWC '19*.

2019 [29] **Vincent Becker**, Felix Rauchenstein, Gábor Sörös. Connecting and Controlling Appliances through Wearable Augmented Reality. *Augmented Human Research, 2019, Springer*.

2019 [30] **Vincent Becker**, Felix Rauchenstein, Gábor Sörös. Investigating Universal Appliance Control through Wearable Augmented Reality. *AH '19*.

2018 [28] **Vincent Becker**, Pietro Oldrati, Liliana Barrios, Gábor Sörös. TouchSense: Classifying Finger Touches and Measuring their Force with an Electromyography Armband. *ISWC 2018*.

2018 [26] **Vincent Becker**, Wilhelm Kleiminger, Vlad C. Coroamă, Friedemann Mattern. Automatically Estimating the Savings Potential of Occupancy-based Heating Strategies. *Energy Informatics 2018*.

2018 [21] **Vincent Becker**. Augmented Humans Interacting with an Augmented World. *MobileHCI '18 Doctoral Consortium*.

2018 [27] **Vincent Becker**, Pietro Oldrati, Liliana Barrios, Gábor Sörös. TouchSense: Classifying and Measuring the Force of Finger Touches with an Electromyography Armband [poster]. *AH '18*.

2017 [25] **Vincent Becker**, Wilhelm Kleiminger. Exploring zero-training algorithms for occupancy detection based on smart meter measurements. *Computer Science - Research and Development, 2017, Springer*.

2017 [22] **Vincent Becker**, Mihai Bâce, Gábor Sörös. Wearable machine learning for recognizing and controlling smart devices. *MobileHCI '17 Workshop*.

2017 [14] Mihai Bâce, Philippe Schlattner, **Vincent Becker**, Gábor Sörös. Facilitating Object Detection and Recognition through Eye Gaze. *MobileHCI '17 Workshop*.