PeriSense: Ring-Based Multi-Finger Gesture Interaction Utilizing Capacitive Proximity Sensing

vorgelegt von
Dipl.-Ing.
Mathias Wilhelm

an der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuß:

Vorsitzender: Prof. Dr. Oliver Brock (TU Berlin)
Gutachter: Prof. Dr. Dr. h.c. Sahin Albayrak (TU Berlin)
Gutachter: Prof. Dr. Paul Lukowicz (DFKI Kaiserlautern/TU Kaiserslautern)
Gutachter: Prof. Dr.-Ing. Dr. h.c. Michael Weyrich (Universität Stuttgart)

Tag der wissenschaftlichen Aussprache: 18. Oktober 2022

Berlin 2022

Zusammenfassung

Wir sind mittlerweile von einer stetig zunehmenden Anzahl an vernetzen Dingen, dem sogenannten Internet der Dinge (IoT), umgeben, ohne es oftmals überhaupt zu bemerkten. Dabei gibt es keinen Lebensbereich mehr, der davon ausgenommen ist. Sei es zum Beispiel das Smart Home, das Auto, oder das vernetzte Arbeitsumfeld. Die Interaktion mit diesen Dingen ist jedoch noch herausfordernd. Die meisten Objekte besitzen eine dedizierte Benutzerschnittstelle, welche nur selten interoperabel ist. Insbesondere Domänenübergreifende Interaktionsmöglichkeiten sind rar. Die Interaktion im Kontext von IoT benötigt daher Interaktionsparadigmen, mit allgegenwärtigen Interaktionsgeräten, die nahtlos im Alltag der Nutzer präsent und nicht an bestimmte Dinge gebunden sind.

Sprachassistenten stellen eine natürliche und intuitive Art der Interaktion mit IoT dar. Jedoch eignen sie sich nicht für alle Interaktionen, Umgebungen und Situationen. Fingergesteninteraktion stellt eine vielversprechende Möglichkeit der Interaktion im IoT-Kontext dar. Sie sind vielfältig und aussagekräftig sowie unauffällig und intuitiv auszuführen. Die Erkennung kann mittels Wearables überall und jeder Zeit erfolgen. Ringe stellen hierbei ein vielversprechendes Wearable dar. Sie werden direkt auf dem Finger getragen und können somit direkt die Fingerbewegung erfassen. Allerdings können sie in der Regel nur die Bewegung eines Fingers erfassen.

In dieser Dissertation wird ein Ring vorgestellt, der ein breites Spektrum an Interaktionen, die mehrere Finger einbeziehen, ermöglicht. Der Ring nutzt dazu kapazitive Sensoren, um die Abstände zu anderen Fingern zu messen. Ein Prototyp, PeriSense genannt, wurde entwickelt und für die Evaluierung verwendet. Diese Dissertation fokussiert auf die Evaluierung der Fähigkeiten und Grenzen von PeriSense. Hierzu wurde neben den technischen Eigenschaften auch der statische und dynamische Interaktionsraum ermittelt. Zusätzlich wurde auch der Interaktionsraum um PeriSense herum evaluiert. Anschließend wurde ein Algorithmus zum Finger Tracking entwickelt, der die kapazitiven Sensormesswerte auf Fingerwinkel abbildet. Mindestens die Bewegung von drei Fingern kann verfolgt werden, und bei vordefinierten natürlichen Bewegungen können sogar die Winkel aller fünf Finger geschätzt werden.

Abschließend werden vier verschiedene implementierte Anwendungen mit PeriSense vorgestellt. Für die Implementierung dieser Anwendungen wurde ein Framework entwickelt. Es ermöglicht die Erstellung von komplexen Interaktionspipelines ohne Programmierung über eine JSON-basierte Konfigurationsdatei.

Insgesamt zeigen die Evaluierungsergebnisse, dass die Verwendung von kapazitiven Sensoren die Erkennung eines breiten Interaktionsraumes unter Einbeziehung mehrerer Finger ermöglicht, wie er von anderen Geräten wie zum Beispiel Kamera-basierten Ansätzen oder Datenhandschuhen bekannt ist. Folglich kann PeriSense ein unaufdringliches Interaktionsgerät darstellen, welches während des gesamten Tagesablaufs zur Verfügung steht.

Abstract

Without even noticing it, we are surrounded by a growing number of networked things, the so-called Internet of Things (IoT). Almost all our life areas are included, such as in the smart home, car, or work. However, interacting with these things is still challenging. Most things provide a dedicated user interface, which is rarely interoperable. In particular, cross-domain interaction possibilities are rare. Interaction with the IoT thus requires interaction paradigms with ubiquitous interaction devices that are seamlessly present throughout the users' daily lives and not bound to any specific things.

Speech assistants are a natural and intuitive way to interact with IoT. However, they are not suitable for all interactions, environments, and situations. Finger gesture interaction represents a promising way of interacting in the IoT context. They are versatile and meaningful, as well as unobtrusive and intuitive to execute. Furthermore, recognition can be achieved anywhere and anytime using wearables. Rings represent a promising wearable in this regard. They are worn directly on the finger and can thus directly detect finger movement. However, they can usually only detect the movement of one finger.

This dissertation presents a ring enabling a broad range of multi-finger interactions. It utilizes capacitive sensing to measure the approximate distances to other fingers. A prototype, PeriSense called, was developed and used for the evaluation. This dissertation focuses on evaluating the capabilities and limitations of PeriSense in the application of finger gesture recognition. For this purpose, besides the technical properties, the static and dynamic interaction space was determined. Additionally, the around-device interaction space was also evaluated. Subsequently, a finger tracking algorithm mapping the capacitive sensor readings to finger angles was developed. As a result, at least the motion of three fingers can be tracked, and in the case of predefined natural movements, all five fingers' angles can be estimated.

Finally, four different implemented applications using PeriSense are presented. For the implementation of these applications, a framework was developed. This framework allows the creation of complex interaction pipelines without programming via a JSON-based configuration file.

Overall, the evaluation results reveal that using capacitive sensing enables a broad interaction space involving multiple fingers, as known from other devices like camera-based approaches or data gloves. Consequently, PeriSense would potentially be an unobtrusive interaction device that is available throughout the daily routine.

Acknowledgments

I want to thank my principal adviser Prof. Dr. Dr. h.c. Sahin Albayrak for the great support and allowing me to conduct my research independently. I would also like to thank the other committee members for their advice and critique to improve my work.

Finally, I want to thank all colleagues at the DAI-Labor for their valuable support. It was a great pleasure to work with every single one of them during the last years. Especially my teammates from the CC SAS contributed a lot in this regard.

My main research has been conducted within the German Research Foundation (DFG) funded project "Erkennung von Mehrfingergesten mit einem Fingerring: Ubiquitäre Interaktion für das Internet of Things" [1]. I would like to thank the German Research Foundation for funding this project and thus enabling my research.

[1] Project website: https://gepris.dfg.de/gepris/projekt/320971279 (accessed on 04.01.2022)

Contents

List of Tables

List of Figures

Declaration of Collaboration

Scientific work is always the result of constant exchange and discussion of ideas, concepts or results, whether with colleagues in the kitchen, through publications or at conferences. This is self-evident. However, this dissertation contains parts that have been actively contributed by Daniel Krakowczyk, who assisted me as a student assistant. He was significantly involved in the production of the prototypes, in particular in the following processes: in circuit design, PCB manufacturing, board assembling, design of the 3D case model, ring assembling, and firmware. The requirements for the hardware, resulting from the findings of the (user) tests and evaluations, originated from me. The implementation of the requirements and the concepts were always discussed and coordinated together. The hardware itself represents an engineering contribution, but not a scientific contribution. Capacitive sensing will not be further developed within this work. This work studies the suitability of capacitive sensing as a technology for multi-finger interaction with a ring. Therefore standard hardware components are used. However, since the ring hardware is a mandatory component for this research, the conception and implementation of the hardware are also described to make the work complete and self-evident. This is done in chapter 5.

All other parts and contributions in this dissertation are from me. For example, the basic idea and conception, the questions, the investigations, the evaluations with the users and tests, the PC-side implementations, evaluations, the implementation of the applications, and so on are entirely from me and were independently implemented by me.

List of Publications

This section lists my publications for which I am an author and which are incorporated in this dissertation.

M. Wilhelm, F. Trollmann, D. Krakowczyk, S. Albayrak: *eRing: Multiple Finger Gesture Recognition with one Ring Using an Electric Field*. In: Proceedings of iWOAR, 2015. https://doi.org/10.1145/2790044.2790047

This publication presents a feasibility study of the basic idea underlying this dissertation. Parts of this preliminary work are presented in the state-of-the-art chapter (Section 3.4) and selectively in the concept part (Section 5.1).

M. Wilhelm, D. Krakowczyk, and S. Albayrak: *PeriSense: Ring-Based Multi-Finger Gesture Interaction Utilizing Capacitive Proximity Sensing*. Sensors, vol. 20, no. 14, p. 3990, Jan. 2020. https://doi.org/10.3390/s20143990

This publication is the primary publication and can be found in many sections of the dissertation. It particularly, but not exclusively, incorporates in the concept and implementation part (Sections 5.1 and 5.2.1) and the sections on the evaluation of the dynamic and around-device interaction space (Sections 6.3 and 6.4).

M. Wilhelm, J.-P. Lechler, D. Krakowczyk, S. Albayrak: *Ring-based Finger Tracking Using Capacitive Sensors and Long Short-Term Memory*. In: 25th International Conference on Intelligent User Interfaces (IUI '20), March 17-20, 2020, Cagliari, Italy. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3377325.337753

This publication presenting a ring-based finger tracking algorithm contributes to the finger tracking section (Section 6.5) in this publication.

M. Wilhelm, J.-P. Lechler, D. Krakowczyk, S. Albayrak: *Demonstration of Finger Tracking Using Capacitive Sensing with a Ring*. In: 25th International Conference on Intelligent User Interfaces Companion (IUI '20 Companion), March 17-20, 2020, Cagliari, Italy. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3379336.338147

This publication relates to the previous one and describes a showcase of the finger tracking algorithm. In the finger tracking section (Section 6.5), this showcase is briefly described.

M. Montebaur, M. Wilhelm, A. Hessler, S. Albayrak: *A Gesture Control System for Drones Used with Special Operations Forces*. In: Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction (HRI '20 Companion), March 23-26, 2020, Cambridge, United Kingdom. ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3371382.3378206

This publication describes an application for this dissertation, which is sketched in Section 7.4.

C.-E. Hrabia, K. Wolf, M. Wilhelm: *Whole Hand Modeling Using 8 Wearable Sensors: Biomechanics for Hand Pose Prediction*. In: Proceedings of the 4th Augmented Human International Conference (AH'13), March, 2013, ACM, New York, NY, USA, 21–28. https://doi.org/10.1145/2459236.2459241

This publication summarize C.-E. Hrabia's co-mentored Master thesis. The proposed finger angle approximation is applied for the finger tracking algorithm in Section 6.5.

List of Supervised Theses

This section lists my mentored student theses, which have been the basis of some of this dissertation's sections.

J.-P. Lechler: *Ring-based Ringer Tracking with Capacitive Sensors Using Neural Networks*. Bachelor Thesis, Technische Universität Berlin, 2018

J.-P. Lechler studied and compared in his Bachelor thesis different neural network architectures mapping the capacitive sensor readings to the finger angles. Based on these results, I developed the finger tracking algorithm proposed in Section 6.5.

M. Montebaur: *A Gesture Control System for Drones Used with Special Operations Forces*. Bachelor Thesis, Technische Universität Berlin, 2019

M. Montebaur utilized the ring developed in this dissertation as controller for autonomous operating drones in his Bachelor thesis. This application is sketched in Section 7.4.

B. Balfagón Ortiga: *Supporting Musical Midi Software Through Gesture Interaction for a Natural and Immersive to Use Interface*. Bachelor Thesis, Technische Universität Berlin, 2016

B. Balfagón Ortiga conducted a user study determining the requirements for a ring-based MIDI-software input controller in his Bachelor thesis. Based on these results, he developed a concept and implemented it prototypically using the ring developed in this dissertation. This application is sketched in Section 7.5.

C.-E. Hrabia: *Entwicklung einer Anwendung zur Erfassung, Analyse und Visualisierung sensor-basierter (Beschleunigungs-, Kreisel- und Magnetfeldsensoren) Handbewegungsdaten zu Zwecken der Erforschung von Fingergesten*. Master Thesis, Technische Universität Berlin, 2012

C.-E. Hrabia developed a flexibly configurable data glove based on motion sensors. Additionally, he developed a graphical tool for recording, analyzing, and

visualizing sensor values from the data glove. Utilizing this tool, he studied and determined approximations for finger angles. Parts of this approximations are used for the finger tracking algorithm in this dissertation (see Section 6.5.1). Also, the 3D hand model implementation was utilized to visualize the ring-based finger tracking.

C. Lackerschmid: *Entwicklung eines Gestenspotter für PeriSense.* Bachelor Thesis, Technische Universität Berlin, 2018

C. Lackerschmid studied different low memory and time complexity features and algorithms for gesture spotting on microprocessors. This work contributes to future work (see Section 8.2) and is implemented on the next generation of PeriSense rings.

1. Introduction

Marc Weiser's vision of ubiquitous computing has already (Weiser, 1999) mainly been fulfilled for many years in the form of the Internet of Things (IoT). We are already surrounded by numerous connected things, often without our noticing. These things consist of a mixture of hardware, software, data, and services aiming to support and assist humans without being obtrusive and distracting or interrupting people from performing a task. Typical applications are self-quantification, connected health, smart home control and monitoring, infrastructure management, industry 4.0 (Talkhestani and Weyrich, 2020; Drews and Weyrich, 1997), and many more. It concerns all domains in our life. Increasingly, the different domains and applications are interconnected so that the user can move and interact seamlessly between these domains. Interaction with the IoT thus requires interaction paradigms with ubiquitous interaction devices that are seamlessly present throughout the users' daily lives and not bound to any specific things.

1.1. Motivation

When interacting with IoT, the smartphone often represents the central interface between the user and the things. Although the smartphone is ubiquitous and seamlessly present, interaction using a smartphone is not natural and intuitive. It requires too many interaction steps and cognitive attention. In addition to smartphones, more and more speech-based assistants are establishing as interaction modalities for IoT. These are mostly available ubiquitously and seamlessly. Moreover, voice-based interaction is natural and intuitive. However, voice-based interaction is not suitable for all situations, such as in public spaces, in crowds, or during a conversation with another person.

Gesture interaction represents a promising modality. Gestures are naturally used in inter-human communication. They also represent a promising modality for interaction with the IoT as they can communicate spatial relations referring to the IoT's physical side. In particular, the use of micro finger gestures, which can be performed easily and inconspicuously, represents a promising interaction. Micro gestures can be performed with low cognitive load and, thus, rather unconsciously (Wolf et al., 2011; Sharma et al., 2019). Since they consist of only small and inconspicuous finger movements, other people in the environment cannot perceive these interactions.

Rings equipped with sensors are a promising basis for ubiquitous gesture detection. Rings are a widely used and accepted article of jewelry in our culture, carrying symbolic and mythical meanings. Due to their small size and form, they can be worn all day without being obtrusive or drawing much attention. Therefore, equipping a ring with sensors can realize a ubiquitous and unobtrusive gesture interaction device. Therefore it is no surprise that sensor rings gain more and more interest in the human-computer interaction community (Shilkrot et al., 2015).

As stated by Shilkrot et al. (Shilkrot et al., 2015) for rings and Lee et al. (Lee and Hui, 2018) in general for wearable interaction devices, current approaches of wearable gesture interaction devices lack a rich range of interactions, which limits the variety of applications. This is due to the fact that rings are only able to sense the motion or the bend angle of a single finger. Possible solutions to this problem are the use of multiple rings (Liang et al., 2021; Fukumoto and Tonomura, 1997) or rings spanning multiple fingers (TapRing, 2017). However, both reduce the obtrusiveness and comfort and thus the suitability for daily use.

This work presents a ring enabling a broad range of multi-finger interactions. It utilizes capacitive sensing to measure the approximate distances to other fingers. A prototype, PeriSense called, was developed and used for the evaluation. This dissertation focuses on evaluating the capabilities and limitations of PeriSense in the application of finger gesture recognition. It is expected that using capacitive sensing enables a broad interaction space involving multiple fingers, as known from other devices like camera-based approaches or data gloves. Consequently, PeriSense would potentially be an unobtrusive interaction device that is available throughout the daily routine.

In the following, the concrete contributions of this work are presented (Section 1.2) and it is described how they related to the thesis structure (Section 1.3).

1.2. Contributions

This thesis develops and evaluates a concept for a ubiquitous interaction device that can be used seamlessly for different applications throughout the day. This dissertation aims to evaluate the capabilities and limits of capacitive proximity sensing for enabling multiple finger gesture recognition with a ring. In the following, the resulting contributions gained from this work are provided:

C1 First main contribution is a concept for a ring utilizing capacitive sensing to enable multiple finger interaction.

C2: A self-contained ring prototype implementing the concept of a ring utilizing capacitive sensing to enable multiple finger interaction is developed and described. This prototype is called PeriSense and is used for the evaluation.

Based on the results of the evaluations, the user feedback, and the experiences made, knowledge is gained on how to improve this prototype towards a final prototype.

C3: The technical evaluation of PeriSense will give insides into the properties of the sensors and their resolution at different distances. Furthermore, knowledge is gained about the susceptibility of capacitive fields to interference in everyday life. Finally, these findings can be applied in general to the design of capacitive sensing-based interaction devices.

C4: The evaluation determines the interaction space reading static, dynamic, and around-device interactions. This allows the derivation of gesture sets and interaction flows which are possible with PeriSense. It also helps to understand the limitations in the recognition of specific interactions. This generic evaluation prevents PeriSense from being tied to a specific application by a dedicated interaction design. Instead, it allows interaction designers to create their interaction applications. In addition to evaluating the interaction space, some diverse applications that have been implemented with PeriSense are presented.

C5: A finger tracking algorithm is presented that can map capacitive sensor data to finger angles. Thus, PeriSense represents a ubiquitous finger tracking device.

C6: A framework is presented allowing the creation of interaction pipelines without programming. This saves time and eliminates the need to program complex interactive sequences. It also reduces the necessary knowledge about interaction flows, gesture recognition, and machine learning. The framework has various interfaces through which the interaction pipelines can be quickly and flexibly integrated into applications. This allows the very fast and flexible creation of interactive applications with PeriSense. The framework is designed modular and flexible so that it would be easily possible to integrate further interaction devices into the framework.

1.3. Structure

In the following, the thesis's structure is described.

Chapter 2 presents the problem analysis consisting of the research challenges and the research questions to be answered.

Chapter 3 illustrates the current state of the art in the related areas, providing a general overview of wearable gesture interaction devices ranging from camera-based approaches over wrist band-based approaches to finger-worn wearable enabling finger interactions. The chapter ends with a presentation of eRing, a pre-

liminary work studying the general feasibility of the underlying concept of multiple finger interaction with a ring using capacitive sensing.

Chapter 4 introduces the fundamental concepts of capacitive sensing and gives a brief overview of applications in the field on human-computer interaction.

Chapter 5 presents the concept of multiple finger interaction with a ring using capacitive sensing. Also, it introduces PeriSense, a prototypical implementation of the concept.

Chapter 6 presents the comprehensive evaluation of PeriSense. It consists of the technical evaluation, the determination of the different interaction spaces, and the finger tracking algorithm.

Chapter 7 presents a generic framework allowing the creation of interaction pipelines without the need for programming. Additionally, it describes four implemented example applications using PeriSense.

Chapter 8 concludes this work with a summary and outlook on future work.

Appendix A shows the intermediate prototypes that led to PeriSense. It also gives a first outlook on the successor developed based on the findings in this thesis.

Appendix B presents all schematics, layouts, and components of PeriSense.

Appendix C presents an example interaction pipeline using the PeriSensePy framework.

2. Problem Analysis

This chapter describes the challenges to be overcome (Section 2.1) and the research questions to be answered in this dissertation (Section 2.2).

2.1. Problem Statement

Interaction in the context of IoT is still challenging. The things to interact with are partially hidden from the user and the variety of things is large. The variety of different use cases and application domains is also significant. At the same time, however, the transition between applications and domains is almost seamless. The diversity of applications makes it difficult to create a uniform interaction concept that can map all specific actions. Constantly changing the interaction device is not practical, so an interaction device is needed that is available at all times and allows a wide range of interactions. Studies revealed that users desire an unobtrusive wearable in the form of jewelry that covers multiple domains and serves as an everyday companion (Fortmann et al., 2015; Ledger, 2014). Overall, the challenges confronted in this thesis are:

- *Ubiquity*: In order to be a full-day companion, the interaction device should work independently of location, time, domain, and other external factors, e.g., illumination or crowded sceneries.
- *Unobtrusiveness*: The interaction device should not obstruct users' daily routine or draw unwanted attention. Ideally, the device should disappear so that the users forget that they wear an interaction device.
- *Richness of interaction set*: The gesture device should be able to distinguish between a large number of different gestures in order to be applicable for as many things, applications, and domains as possible (Shilkrot et al., 2015; Lee and Hui, 2018).

2.2. Research Statement

Rings have good prerequisites to meet the challenges defined above. Rings are usable independent of time and location, and they can become invisible to the user. The drawback is that they lack a rich interaction set. This dissertation aims to

extend the ring's capability to recognize multiple finger gestures. For this purpose, capacitive sensing is utilized. Consequently, the interaction space is to determine and to study. In detail, the following questions are to be studied:

Q1 *Which technical properties can be achieved using standard consumer electronic components?* These properties involve the determination of the resolution at different distances and the study of the influence of environmental noise on the capacitive measurement.

Q2 *Which gestures and interactions are possible and where are the limits?* To answer this question, it is aimed to determine PeriSense's interaction space for different interactions. For this purpose, there are the following sub-questions:

- Q2-1 *What is the static interaction space?* Which fingers are in range and which not in the case of static gestures such as postures? Which kind of postures are possible, and which implications exist?
- Q2-2 *What is the dynamic interaction space?* Which finger movements are in range and which not? Which kind of gestures are possible, and which implications exist?
- Q2-3 *What is the around-device interaction space?* What kind of gestures can be performed nearby the ring? Which gestures are distinguishable and which not? Which kind of gestures are possible, and which implications exist?

Q3 *Can the interaction space increased by additional sensor data information?* The PeriSense prototype contains also a motion sensor. Does this motion sensor contribute to the extension of the different interaction spaces?

Q4 *Is it possible to map the ring data to a hand model in order to determine the angles of each finger joint?* If it is possible, how accurate is the determination of the angles?

The main focus of this dissertation is the evaluation of PeriSense's interaction space. This work aims not to extend the capacitive sensing technology. To implement the concept, commercial state of the art components are used.

3. Related Work

Wearable interaction devices undergo a continuously increasing popularity in the HCI community. This led to a wide variety of different wearables. This chapter presents and discusses related work in the context of wearables enabling finger interactions. It starts with an overview of finger-worn wearables and extends to arm and body-worn wearables in general. It continues to give an overview of wearables enabling finger tracking. Finally, preliminary work conducted prior to this dissertation is presented. This work studied the general feasibility of using capacitive sensing for multiple finger interaction with a ring.

3.1. Finger-worn Wearables Enabling Finger Interaction

Probably the largest category of finger worn interaction devices is rings. Most of them are equipped with motion sensors such as accelerometer (to sense acceleration), gyroscope (to sense the change of orientation) and magnetometer (to sense the relative orientation with respect to the earth) in order to sense gestures and finger taps (Liang et al., 2021; Younas et al., 2020; Gupta et al., 2019; Ens et al., 2016; Fukumoto and Suenaga, 1994; Gummeson et al., 2014; Hrabia et al., 2013; Jing et al., 2013; Ketabdar et al., 2012; Nirjon et al., 2015; Wolf et al., 2013, e.g.). This allows the recognition of a wide range of dynamic gestures and interactions in mid-air and on surfaces. However, usually, only the movement of one finger or the whole hand is captured. DualRing (Liang et al., 2021) utilizes two rings equipped with motion sensors worn on the thumb and index finder. To capture all individual finger's movement, all fingers must be equipped with a ring (Hrabia et al., 2013, e.g.). If a ring is combined with a sensor wristband and the sensor data is fused, as in the case of FinGTrAC (Liu et al., 2020) and WRIST (Yeo et al., 2019a), then the interaction variety can be expanded by including the orientation of the hand or finger in relation to the wrist.

Another common type of sensors are magnetometers sensing the influence of permanent magnets in the close-proximity environment, such as uTrack (Chen et al., 2013). Two magnetometers are worn on the ring finger, enabling 3D input by wearing a magnet on the thumb. A similar configuration is proposed in (Reyes et al., 2018), where a magnet is worn on a thumb ring and a magnetometer of a smartwatch is used to sense the extension and reposition of the thumb in order to control smartwatch applications. A similar approach is used by Ma et al. (Ma et al., 2011)

where a magnet placed on the index finger is tracked by magnetometers mounted on a wrist band. In contrast to these approaches, in (Simmons and Welsh, 2015, 2014) a magnetometer is worn on each finger and the magnet is placed on the back of the hand. This configuration allows accurate finger tracking but it requires equipping the whole hand. Parizi et al. (Parizi et al., 2019) follows a similar approach with AuraRing, where instead of a magnet a coil is embedded in the ring. Several coil sensors around the wrist measure the magnetic field, from which the finger position is determined. Nenya (Ashbrook et al., 2011) and PairRing (Chung et al., 2018) use magnets in the ring and magnetometers mounted on a wrist band to recognize ring rotations along the finger axis. MagTouch (Park et al., 2020) and FingMag (Park and Lee, 2019) use a Ring with a magnet to increase the touch precision on smartwatch touch screens. The magnetometer of the smartwatch is used to determine the finger position through the ring's magnet.

Zhang et al. (Zhang et al., 2011) uses a piezo microphone to recognize slide and tap gestures performed on a surface. The same approach is used by FingerSound (Zhang et al., 2017a) and FingOrbit (Zhang et al., 2017b) enabling unistroke gestures drawn with the thumb on the palm. These piezo microphone equipped rings are worn on the thumb and recording the sound when the thumb is moving over the palm. Additionally, a gyroscope is used to track the directional changes of the thumb. This configuration allows a high accuracy recognition of Graffiti-style letters.

Many ring devices provide touch areas for binary input. OctaRing (Lim et al., 2016) for example, consists of eight touch areas for pressure-sensitive multi-touch input enabling complex input patterns. Thumb-In-Motion (Boldu et al., 2018a) utilizes a capacitive touch-matrix on a ring in order to enable thumb slide and tap gestures recognition on the ring.

ThermalRing (Zhang et al., 2020) uses a low-resolution thermal camera mounted on a ring. The ring is worn on the index finger of on hand while the camera points sidewards. On the index fingertip of the other hand, there is placed a thermal reflective tag. The ring has a thermal source emanating heat. If the tag is in the camera's field of view, the heat is reflected. This reflection is visible as a white track-able spot in the camera image. This tag can also be placed on objects allowing on-device interactions.

Currently, there exist only two rings sensing the movement of multiple fingers: TypingRing and CyclopsRing. TypingRing (Nirjon et al., 2015) detects finger taps on a surface in order to enable text input. It uses proximity sensors in order to detect if the neighboring finger is next to the ring or not. CyclopsRing (Chan et al., 2015) utilizes a fish-eye camera in order to enable a wide range of finger interactions. The fish-eye camera point downwards capturing all fingers. However, the camera lacks under fast and intense changes in illumination and the recognition has to deal with occlusions and background noise. Furthermore, it is much more challenging to

implement a resource-friendly image transmission and the corresponding gesture recognition approach on a mobile phone, not to mention on the ring itself.

ThumbTrack (Sun et al., 2021) uses an array of infrared-based proximity sensors placed around a ring. Conceptually, this concept is closest to that of this dissertation. However, ThumbTrack is only evaluated for recognizing thumb-based pinch gestures and not for any further finger interactions. Also, the ThumbTrack approach lacks of a self-contained concept for a implementation in the wild.

There exist a further couple of wearable finger interaction devices beside rings. Most notable are nail covers (Gu et al., 2020; Shi et al., 2020; Hsiu et al., 2016; Kao et al., 2015). They have similar capabilities such as ubiquity and unobtrusiveness. While Nail+ (Hsiu et al., 2016) can only detect different force touch interactions, NailIO (Kao et al., 2015) can distinguish on-nail finger swipe gestures. Shi et al. (Shi et al., 2020) use a motion sensor mounted on the index finger's nail to enable drawing gestures on a surface. QwertyRing (Gu et al., 2020) uses a similar approach for a virtual keyboard application.

Another approach is TIMMi (Yoon et al., 2015) a textile finger worn input devices. It is worn on the index finger to measure its finger bend and recognizes touch events with the thumb on TIMMi itself by the means of a conductive elastomer. A similar but more flexible approach is proposed by FabHandWear (Paredes et al., 2021).

Rings equipped with sensors are a promising basis for ubiquitous gesture detection. Due to their small size and form, they can be worn all day without being obtrusive or drawing much attention. Consequently, they depict a ubiquitous and unobtrusive gesture interaction device. Despite rings offer many interaction possibilities, most rings can only detect a particular interaction set, and the movement detection is limited to one finger or the whole hand. Multi-ring concepts can provide a wide range of interactions. However, they lack comfort since all fingers must be equipped with a ring as well as an energy supply and a resource-friendly signal processing concept for multiple rings is needed. Current single ring approaches cannot detect, for example, whether all fingers are formed into a fist or only one or two fingers are bent. Only CyclopsRing can detect complex multi-finger gestures. However, it uses a camera. This prevents objects from being held in hand during the interaction. Also, the power supply over several hours, avoiding the lens from becoming dirty and extreme illumination situations are challenging. In Addition, it can cause privacy concerns for the user. In summary, rings bring good prerequisites to be an everyday companion. However, they usually lack in providing a broad interaction space.

3.2. Arm and Body Worn Wearables Enabling Finger Interaction

Besides rings also other wearables can enable the detection of finger interactions. An obvious wearable enabling multiple finger interaction is a data glove (Lee et al., 2019; Hsieh et al., 2019; Nguyen et al., 2019; Takada et al., 2019; Peshock et al., 2014; Murao, 2015; Hsieh et al., 2016; Dipietro et al., 2008, e.g.). It allows the equipment of a multitude of sensors, enabling it to detect and track the finger motions and touch interactions very precisely. However, sensor gloves suffer in tactile feedback and comfort because users must wear them everywhere throughout the day. This is uncomfortable in summer or wet conditions such as sports. Gloves are thus more suitable for particular short-time purposes such as input for virtual reality glasses to manipulate virtual objects or accurate machine control (Drews and Weyrich, 1997, e.g.).

Another often found approach is the usage of a camera to track finger movements. Some approaches mount the camera, for example, on a head-mounted display (Chatain et al., 2020; Dominguez et al., 2006) or on a shoulder (Soliman et al., 2018). However, here, for the interaction the fingers have to be brought into the camera's field of view. Other approaches place a camera on a wrist. Wu et al. (Wu et al., 2020a) place a camera at the top of a wrist band. The gestures are recognized by distinguishing the hand contour caused by the different gestures and postures. Opisthenar (Yeo et al., 2019b) uses a similar approach. Instead a rgb-camera it uses a depth camera that captures overstretching finger postures so that the finger tips are in the camera's field of view. Both approach, however, are limited in the interaction manifold and in case of Opisthenar overstretching finger postures lead fast to fatigue. On the opposite to place the camera on the top of the wrist, UbiHand (Ahmad and Musilek, 2006) and Digits (Kim et al., 2012) use a camera placed downwards capturing the palms inside. This allows accurate tracking of all fingers. FingerTrak (Hu et al., 2020) uses four thermal cameras placed around a wrist band to enable to full hand tracking. RotoWrist (Salemi Parizi et al., 2021) applies a similar approach using infrared light-based sensors to track the wrist's orientation and motion. WristLens (Yeo et al., 2020), Yamato et al. (Yamato et al., 2020), WatchSense (Sridhar et al., 2017), PalmType (Wang et al., 2015a), and PalmGesture (Wang et al., 2015b) use a wrist band mounted camera for the recognition of gestures and interaction performed with the other hand's index finger on the arm or hand surface. Camera-based wearables can enable manifold interactions, precise tracking, and high recognition performance. However, in the context of ubiquitous computing, there are three major drawbacks of these interfaces: occlusion, illumination sensitivity, and a limited field of view (in dependence on the mounting position). In the case of a head or body-mounted camera, for example, the user must bring his limbs into the camera's active field of view, which is in the front of the

user's body in this case. This can be exhausting for the user. Further, such interactions need some space in front of the user's body, making it challenging to gesture, for example, in a crowded space. This interaction style can also make some users feel strange and awkward by performing obtrusive gestures in public. In contrast, wrist-mounted camera solutions allow unobtrusive finger gestures, but they have to deal with occlusion by jackets, gloves, or other clothes and objects or dirt. Despite this, all camera-based approaches have to deal with changing and extreme illumination conditions, such as strong direct sunlight or absence of light, and cluttered background noise. Additionally, the transmission and processing of images still require many resources. Consequently, building a self-sufficient and energy-efficient camera-based wearable is, therefore, still a considerable challenge.

BeamBand (Iravantchi et al., 2019) uses an array of ultrasonic sensors mounted on a wrist band to detect different hand gestures. However, it is confronted with similar problems as the camera-based approaches.

Finger movements causes a displacement of the bones and chords changing the arm contour. FirstVR (Tamaki et al., 2019), Hosono et al. (Hosono et al., 2019), Karasawa et al. (Karasawa et al., 2019), and Fukui et al. (Fukui et al., 2011) use photo reflector sensors to detect changes in arm contour. Furthermore, there are many different methods to measure these changes. Bian et. al(Bian and Lukowicz, 2021), GestureWrist (Rekimoto, 2001), and CapBand (Truong et al., 2018) apply capacitive sensing, WristFlex (Dementyev and Paradiso, 2014) uses pressure sensors, Ortega-Avila et al. (Ortega-Avila et al., 2015) utilizes infrared lights, EchoFlex (McIntosh et al., 2017) applies ultrasound imaging, and Tomo (Zhang and Harrison, 2015) use tomography to measure these arm contour changes. Also, electromyography (EMG) measuring the electrical activity produced by muscles in the forearm can be used to detect and distinguish finger movements as shown, for example, by CAPG-MYO (Dai et al., 2021), Zhang et al. (Zhang et al., 2018), Amma et al. (Amma et al., 2015), Haque et al. (Haque et al., 2015), and Huang et al. (Huang et al., 2015). EMPress (McIntosh et al., 2016) combines arm contour measuring with pressure sensors and muscle activity measurement with EMG sensors to detect finger gestures. Depending on the used technique, the interaction space for these wrist bands ranges from rough hand movements to finger movements. Fine-grained micro gestures are hard to detect. Additionally, most sensing techniques are sensitive regarding noise and changes in displacement. Some also lack comfort since they have to be fixed tight around the arm in order to be able to measure the arm contour and chords displacements.

Noticeable is the capacitive sensing-based wrist band proposed by Bian et. al(Bian and Lukowicz, 2021). It is one of the very few self-sufficient interaction devices meeting challenges like energy consumption and gesture recognition in the wild. Even the deep learning-based gesture recognition is performed on the ring's microprocessor. Most approaches lack a concept for self-sufficient power supply and data

processing in the wild.

Kato (Kato and Takemura, 2016) and Yokota (Yokota and Hashida, 2016) use a high-frequency sound that is induced into the forearm. Hand and finger movements cause a frequency shift of the induced sound which can be applied to recognize finger gestures. SoundCraft (Han et al., 2017) in the opposite uses microphones integrated into a smartwatch to record environmental sounds, e.g., moving the finger over a desk, which is then utilized for hand gesture and interaction recognition. Laput and Harrison (Laput and Harrison, 2019) detect fine grained hand activities using high-speed acceleration data from a Smartwatch. In combination with motion sensors, they are able to detect a wide spectrum of hand and finger gestures. On the other hand, they can only detect broad gestures and no subtle finger movements.

AuraSense (Zhou et al., 2016) enables around-smartwatch interaction by electric field sensing. It enables the detection of finger movements up to $3\,\mathrm{cm}$ around the device. It allows primarily the detection of slide movements. A similar approach is realized by Watanabe et al. (Watanabe and Terada, 2018) applying ultrasound. FingerIO (Nandakumar et al., 2016) generalizes this idea. It transmits an inaudible signal and tracks the echoes of the moving finger with a smartphones' microphone. It works even when the phone is in the pocket. However, the recognition is challenging when the device is moving. Additionally, the computation requires many resources resulting in very high power consumption on the smartphone.

SeeSaw (Wu et al., 2018a) uses the smartwatches' motion sensor to detect arm rotation for smartwatch control. fSense (Buddhika et al., 2019) uses a photoplethysmogram sensor in a smartwatch to recognize some hand gestures.

A special case of wearables are on-skin interfaces consisting of stick-able sensors such as PDMSkin (Röddiger et al., 2020) and Tacttoo (Withana et al., 2018) or implantable interface (Holz et al., 2012). These interfaces are still limited for 2D touch and tap gestures. Similar considerations also apply to smart textiles (Wu et al., 2020b; Yoon et al., 2015) which can additionally also be used to detect finger or arm bends (Wu et al., 2018b; Yoon et al., 2015).

Despite all advantages such as general acceptance, unobtrusiveness and the possibility of integration into smartwatches, the wrist bands cannot sense subtle finger gestures, are partially sensitive to noise and uncomfortable in case of arm contour measurement devices or need still too many resources in case of camera-based devices.

3.3. Finger Tracking Using Wearables

Besides finger gesture recognition, there also exist wearables to track all finger angles continuously. Lee et al. (Lee et al., 2019) developed a glove for virtual reality applications. It tracks the thumb, index finger, and middle finger with high precision using motion and soft sensors. Additionally, it provides haptic feedback measuring

the fingertip contact force. However, the extensive sensor equipment providing this high precision requires a cable for power supply and data transfer. In the context of virtual reality, it is acceptable, but in the context of ubiquitous computing not. Kim et al. (Kim et al., 2016) provides a similar glove to track all fingers with similar precision and haptic. Nassour et al.(Nassour et al., 2020) provides a more everyday use-able glove but with some lower precision. It uses conductive sensors in silicon tubes. This allows a simple adaptation to different glove sizes.

Hrabia et al. (Hrabia et al., 2013) developed a customize-able glove based on motion sensors. Sensors for each finger joint can be added or removed. Missing sensors respectively joint data will be approximated.

FingerTrak (Hu et al., 2020) uses four thermal cameras placed around a wrist band to track 20 finger and hand joints. The joint angles are estimated from finger postures using a deep neural network. The average mean error over all joints is 6.46 degrees. Cluttered background and re-mounting can increase the error to 8.06 degrees. Digits (Kim et al., 2012) tracks all fingers using a depth camera mounted on a wrist. It is placed downwards, capturing the palms inside. For each joint, an average mean error of fewer than 9 degrees is reported.

WU-Hand (Liu et al., 2021a) and Liu et al. (Liu et al., 2021b) enable full finger tracking with a electromyography-based wrist band by utilizing a deep learning approach.

Ma et al. (Ma et al., 2011) and Lyons et al. (Lyons, 2020) place a magnet on the index fingers' nail, which is tracked by magnetometers mounted on a wrist band. This method allows, however, only the tracking of one finger. MagX (Chen et al., 2021) solves this issue by developing an enhanced sensor array allowing a flexible placing of multiple magnets. Simmons et al. (Simmons and Welsh, 2015, 2014) places a magnetometer on each finger segment and a strong magnet at the wrist. This enables high precision tracking. The average mean error over all fingers is 2.55 degrees. The drawback, however, is that all fingers have to be equipped with sensors.

There exists no ring providing finger tracking, not even for just one finger. Ma et al. (Ma et al., 2011) and Lyons et al. (Lyons, 2020) provide the smallest setup equipping one finger and the wrist. All remaining approaches equip all fingers or using a wrist band. It would be possible to track a single finger by just equipping the finger with a soft sensor or conductive sensor. However, this would require a textile tube as a carrier for the sensors. Such a textile, however, would reduce the comfort in everyday usage. Further, there is no device sensing multiple fingers except wrist band-based approaches.

3.4. eRing: A Preliminary Feasibility Study

At the beginning, there was an idea to create electric fields around a ring and to measure their distortions by the fingers' movements. Measuring these disturbances could provide feedback on the finger movement or even current finger position and thus enable gesture recognition. Before the dissertation, the general feasibility of this assumption was examined. For this purpose, a prototype called eRing was developed to explore if it is worth studying the idea in more depth in this dissertation (Wilhelm et al., 2015).

In the following, this section presents a brief summary of this preliminary work and the results.

3.4.1. Concept

Electric fields are omnipresent and act on electric charges, while capacitance is the ability to store this charge. Typically a capacitor is presented as a parallel pair of two metal plate electrodes which constitute an electric field. For electric field sensing, one electrode that builds up a capacitor with its dielectric environment is sufficient. In the case of a nearby moving conductive object the capacitance changes as a consequence. This variable capacitance is therefore a well-fitting way of measuring the proximity of the conductive human skin and is widely used as a non-contact touch and proximity sensor (Baxter, 1996).

To measure the variable capacitance, a simple RC circuit(Figure 3.2) was used for the first prototype. In this operation mode, the changes of the surrounding electric field are sensed by measuring the charging time of the electrode. Setting the send pin to a high value will have a measurable delay on the receive pin due

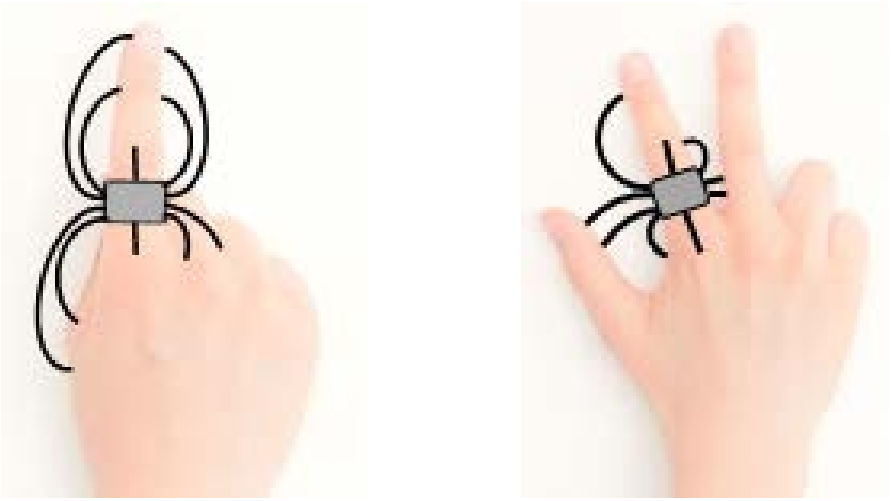


Figure 3.1.: Visualization of the preliminary idea.

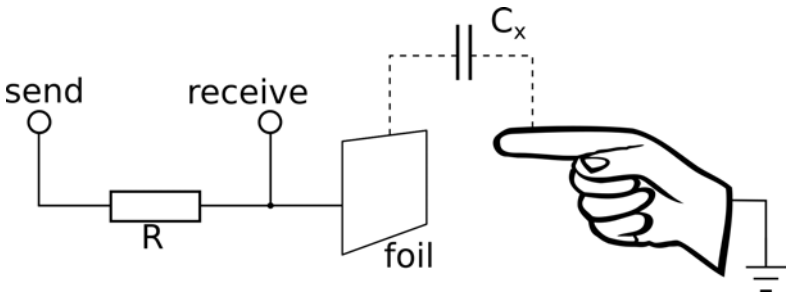


Figure 3.2.: Circuit for a single electrode of eRing.

to the charging of the capacitor. This rise time τ is proportional to the product of resistance and capacitance, which enables the inference of the sought capacity C_x.

Apart from the capacitor electrode the only explicit component is thus the resistor. Low values will result in shorter charging times, while high values will result in a more sensitive response. Hence there's a trade-off between response time and resolution.

3.4.2. Implementation

The constructed prototype called eRing (Figure 3.3) consists of four components. A 3D printed ring provides the body to attach the sensor system. Four copper foils are attached to the ring's lower sides and act as distinctive electrodes for capacitive sensing. To prevent direct contact between skin and electrode, adhesive tape is used as isolation. A small board with four $10\,\mathrm{M}\Omega$ resistors is mounted on top of the ring and serves as the driving circuit for the capacitive measurement. Sitting on its top, an Arduino Nano[1] sends step inputs and measures the respective time delay by utilizing the open source CapSense library written by Paul Badger[2]. The Arduino is connected via USB to a PC for processing and classification. Due to its minimal design the system consumes a total power of maximum $0.2\,\mathrm{W}$. The consumption is almost entirely determined by the micro controller. The sensor driving board consumes $10\,\mu\mathrm{W}$.

3.4.3. Gesture Recognition

In order to classify gestures in the four dimensional sensor data, we use a 1-nearest neighbor (1NN) classifier with lucky time warping (LTW) (Spiegel et al., 2014) as a similarity measure. It has been shown that the 1NN classifier is a good choice for time series classification (Xi et al., 2006), such as gesture recognition. Additionally, it is parameter-free, requires only a small number of training examples, provides

[1] Arduino Nano's website: https://store.arduino.cc/arduino-nano (accessed on 21.03.2021)

[2] CapSense Arduino library: `http://playground.arduino.cc/Main/CapacitiveSensor` (accessed on 21.03.2021)

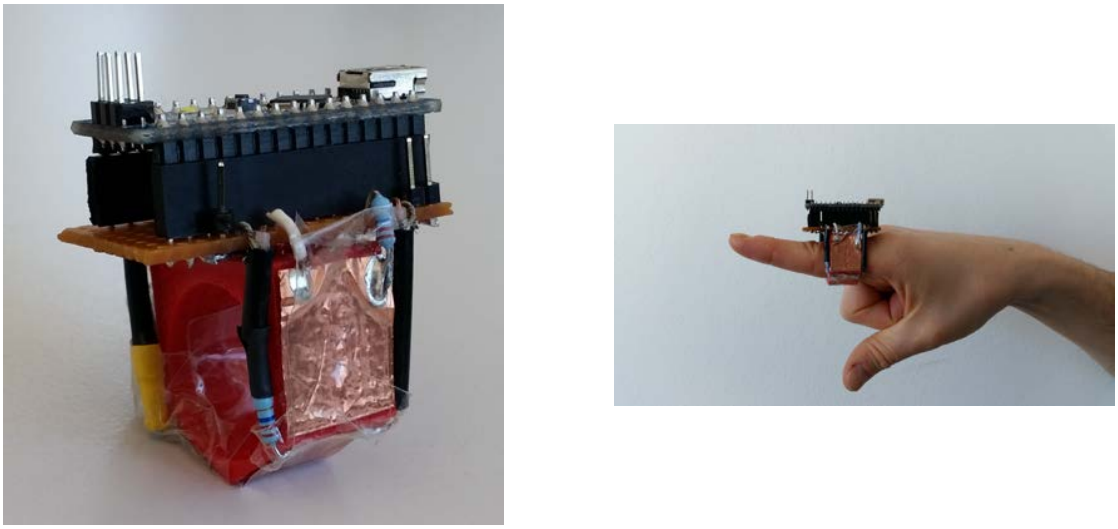


Figure 3.3.: eRing prototype.

easy to comprehend results, and has in connection with LTW a linear time and space complexity. This allows the implementation of the classifier on microcontroller.

Since we obtain a four-dimensional time series from the ring, we have to replace the one-dimensional distance function $d(q_i, c_i)$ in (Spiegel et al., 2014, Eq. 7) with: $d(q_i, c_i) = \sum_{d=1}^{D=4}(q_{i,d}, c_{j,d})^2$, where $q_{i,d}$ is the d^{th} dimension in i^{th} data point in the time series $Q, c_{j,d}$ is the d^{th} dimension in j^{th} data point in the time series C, and D is the number of dimensions. For the prototype presented in this paper, we introduce two classifiers: one for static finger postures and one for dynamic finger gestures. The difference between both is the preprocessing step of the time series. For the dynamic finger gestures, we apply the z-score normalization: $Q_z = \frac{Q-\mu(Q)}{\sigma(Q)}$, where Q is a times series to be normalized, $\mu(Q)$ is the mean of Q, $\sigma(Q)$ is the standard deviation of Q, and Q_z is the normalized time series. This normalization is important because the amplitudes of one gesture class can have strong variations. These variations have a big influence on the similarity measure and can lead to false classifications.

In contrast to static finger postures, there is no time varying pattern but different constant amplitudes. A z-score normalization would eliminate the distinguishability of different posture classes, because after the normalization the posture data is close to the zero-baseline. Accordingly, for dynamic finger gesture recognition we apply the z-score normalization before we classify the gesture with the 1NN classifier and the recognition of static finger gesture is based on the raw sensor data for the 1NN classification.

d	**0**	**4**	**7**	**10**	**14**	**17**
μ	1.5365	1.43	1.4089	1.3954	1.3918	1.39
σ	411	140	79	27	13	6
μ_f	1.6801	1.5420	1.5134	1.5046	1.4996	1.4980
σ_f	876	190	79	32	14	8

Table 3.1.: Mean μ and standard deviation σ values of the 20 runs for sensor 1 over different distances d in approximated millimeter. The index f indicates the signal, which was recorded with the ring worn on the finger.

3.4.4. Evaluation

In order to show the general feasibility of the idea of an electric field-based ubiquitous gesture device, three experiments were conducted to evaluate eRing. In the first experiment, the range of the electric field and the sensitivity were explored. Based on these results, six postures and six dynamic gestures were defined to test the recognition capabilities of the prototype.

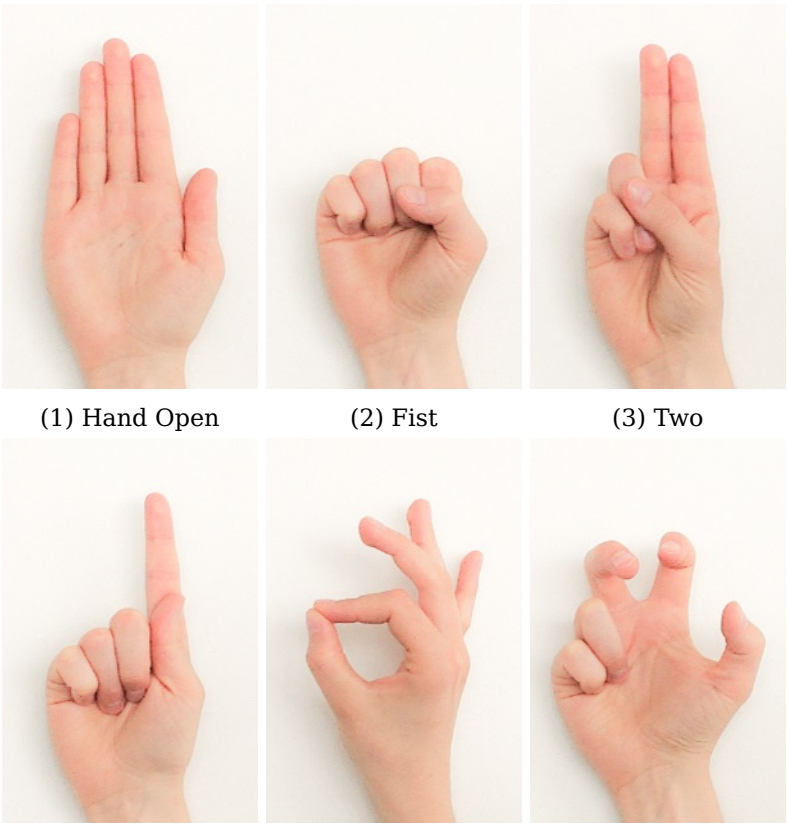
Experiment 1 – Sensitivity and Field Range Test

First, the field range and sensitivity of eRing were tested. For this, a WACOM pen tablet[3] and pen was used to measure the ranges. The ring was fixed on the tablet. The pen was held between thumb and the index finger of the left hand. The index finger was placed at the ring, so that it touched one electrode, and it was moved $2cm$ away from the electrode along a ruler fixed on the tablet while recording the sensed values of the electrode and the position of the pen. To simulate the influence of a finger inside the ring, the ring was placed on the right hand index finger and performed the same procedure as before. The sensor values, the moved distances in pixel ($2cm$ equals about 90 pixel), and the time line were written in a file. Afterwards, the mean and standard deviation of the 20 runs was computed.

Results The experiment revealed that the average field range is about 1cm. The values increase only slightly for distances bigger than about 5mm. For smaller distances clear signal changes can be observed. However, with decreasing distance, the standard deviation increases rapidly. Table 3.1 shows this relationship for sensor 1. It reveals also that the standard deviation increases and the field range decreases, if the ring is worn on the index finger.

Discussion This experiment shows that the electric field of eRing is small. For this reason, only the tracking of neighboring fingers is realistic. Since the effective field range is only about 5mm, the neighboring finger should be close to the ring.

[3]WACOM Intous™3 Graphics Tablet PTZ-1230



(1) Hand Open (2) Fist (3) Two

(4) Pointing (5) Ring (6) Grasp

Figure 3.4.: Finger postures used in experiment 2.

Experiment 2 – Finger Posture Recognition

In a second experiment, it is aimed to show the feasibility of detecting finger postures with eRing. Based on the results of experiment 1, six postures involving the thumb, the index finger, and the middle finger (Figure 3.4) were defined. Due to the limited range of the prototype gestures were chosen that can be distinguished based on the position and distance of thumb and middle finger if the ring is placed on the index finger (cf. Figure 3.3). Fifty examples for each posture were gathered in one session from one user. In total, 300 posture examples were collected. For this process, an evaluation software were used which asked the user to perform a certain posture. The sequence of postures to perform was random and no posture was chosen in sequence. In order to be able to segment properly the postures for the offline classification, the participant labeled the postures with a button installed with a wire on eRing. Every time he moved his finger in the requested position, he pressed the button for at least one second. The four dimensional sensor data, the time line, the requested labels and the button state were written in a file.

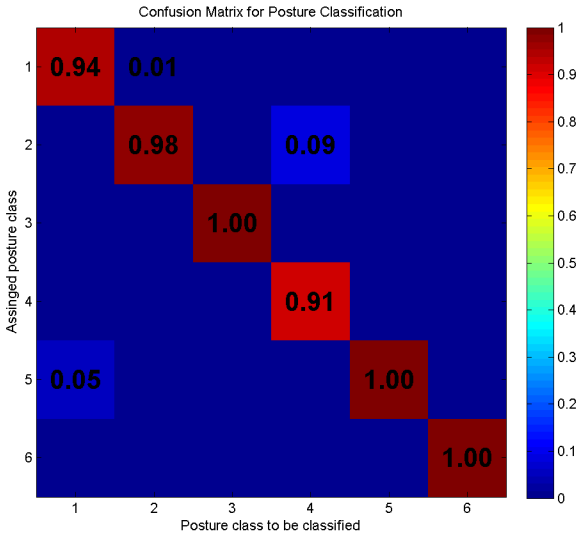
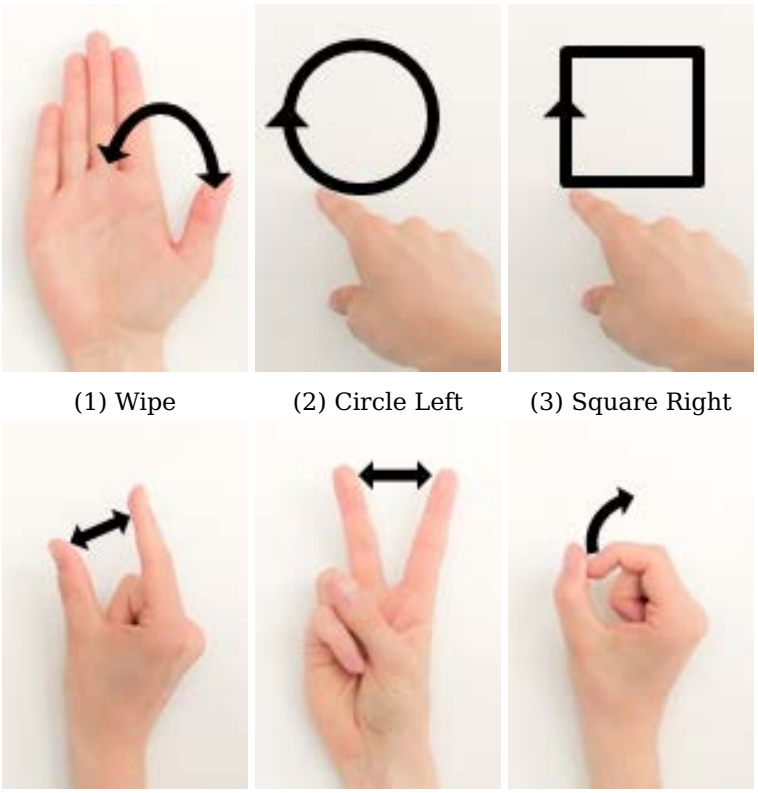


Figure 3.5.: Confusion matrix for experiment 2.

After the recording, the labeled postures were extracted using a script which stored them in a suitable data structure. In order to decrease the influence of the selected training examples and to provide reproducible results, a cross-validation was performed. The data set was split into 10 sequential partitions. Each partition contained 30 training examples. These are 5 examples for each posture, which fits a practical size for online gesture recognition. We took one partition as the training set and the remaining nine were used as test set. This procedure was repeated for each of the 10 partitions. For the classification of the postures, the raw sensor data and a 1NN classifier with lucky time warping (LTW) (Spiegel et al., 2014) as similarity measure was used.

Results The mean error rate of the 10 runs in experiment 2 was 0.027. Figure 3.5 contains the confusion matrix of the experiments where it can be identified which postures were miss-classified. The columns represent the posture classes and the rows represent the class as which the classifier labeled the examples for this posture. The numbers of the column and the row headers correspond to the posture numbers in Figure 3.4. The matrix summarizes the results of all 10 runs and the values were normalized to 1 (matrix value / (45 test examples per posture * 10 runs)). Due to rounding bias, some columns do not sum up to one. The matrix shows that the postures "Two" (3), "Ring" (5), and "Grasp" (6) show the best recognition performance. Some examples of the posture "Fist" (2) were assigned to posture "Hand Open" (1). The postures "Hand Open" (1), "Fist" (2), and "Ring" (5) received some false-assignments by "Fist" (2), "Pointing" (4), and "Hand Open" (1), respectively. Only "Two" (3), "Pointing" (4) and "Grasp" (6) received no false-assignments.



(1) Wipe (2) Circle Left (3) Square Right

(4) Pinch (5) Snap (6) Drop

Figure 3.6.: Finger gestures used in experiment 3.

Discussion The experiment revealed that it is possible to distinguish between finger postures including thumb, index finger, and middle finger with an average of 97% correct classified test examples. Most miss-assignments can be explained by the limited range of the electric field. For example, during the execution of "Hand Open" (1), the thumb was sometimes too far away from the ring and was consequently, classified as "Ring" (5) or "Two" (3). If the fist was not fully closed, posture "Fist" (2) was sometimes classified as "Hand Open" (1). The posture "Pointing" (4) performed worst because it produces a similar signal pattern to the posture "Fist" (2) if the middle finger was in effective field range. Overall, all postures had in average over 90% of correct results and are, thus, suitable for the usage with eRing.

Experiment 3 – Dynamic Finger Gesture Recognition

After the feasibility of eRing to recognize finger postures was evaluated, it is aimed now to show the feasibility to recognize dynamic finger gestures involving thumb, index and middle finger. In this experiment, the recognition of six gestures (Figure 3.6) were tested. The data in this experiment was recorded under the same condi-

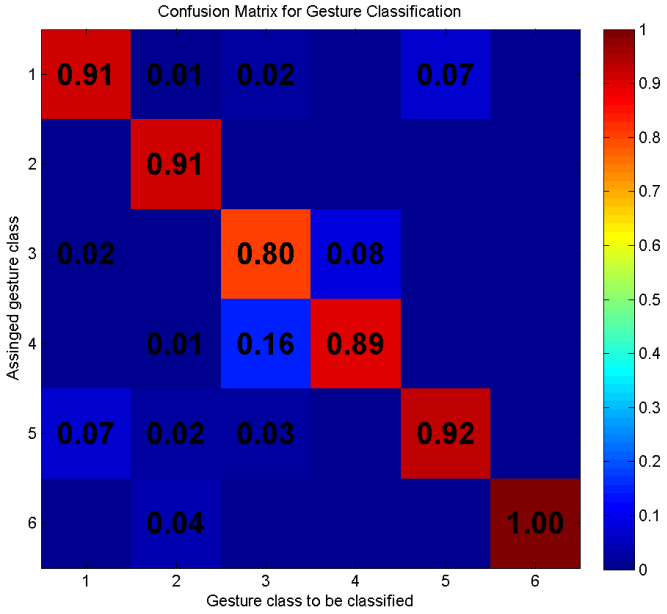


Figure 3.7.: Confusion matrix for experiment 3.

tions as in experiment 2 for the posture recognition. In difference to the posture recognition, a z-score standardization to each gesture example was applied before the classification was run.

Results The results of experiment 3 are mapped into a confusion matrix (Figure 3.7) in the same way as described in the results of experiment 2. It shows a multifarious distribution of miss-classified examples. The average classification rate of each class was 0.91. The outliers are "Drop" (6) with 1.00 and "Square" (3) with 0.80. The overall mean error rate of the 10 runs in experiment 3 is 0.094. The gestures "Circle" (2) and "Square" (3) had the broadest distribution of miss-classified examples. Most miss-classified examples of the gesture "Square" (3) were assigned to "Pinch" (4) and vice versa.

Discussion This experiment showed that eRing is able to detect a broad range of gestures. It enables the recognition of index finger movements (2, 3, 5, 6) as well as of thumb (1, 4) and middle finger (5) movements close to the index finger. The same issue with the limited field range as observed in experiment 2 occurs in context with the dynamic gestures. If the fingers are further away from the ring than they should, the gesture is miss-classified. For example, if the thumb is out of the field range during the "Wipe" (1) gesture, then it is classified as "Snap" (5) or "Square" (3). However, in spite of the lower overall recognition rate in comparison to the postures, all gestures are suitable for the usage with eRing.

3.4.5. Discussion and Limitations

In this preliminary work, a novel ubiquitous gesture interaction device called eRing was proposed and its feasibility evaluated. It consists of a ring equipped with four electrodes spanning an electric field around the finger. Wearing eRing on the index finger, finger postures and gestures involving the thumb, index finger and middle finger can be recognized.

Experiments showed that despite severe limitations in the prototype and no optimization in the detection algorithm the prototype is a promising foundation for distinguishing finger postures and gestures. To overcome limitations in the range of the electric fields a new prototype should be build using an oscillating voltage and a different measuring mode for the variations in the electric field.

4. Capacitive Sensing

Capacitive sensing is a well-studied and widely applied technology. Probably the oldest and most famous example of an application is the Theremin[1] a non-contacting electronic music instrument. Today, it is impossible to imagine our everyday life without this technology. For example, the touchscreens of smartphones are based on the principle of capacitive sensing. Also, many sensors such as accelerometers, which can be found in every smartphone and many other devices today, are based on this technology. This chapter introduces the basic principles of capacitive sensing. It is limited to the slightest necessary details to understand and reconstruct this work. Readers interested in more details in the fundamentals of electrostatics, circuit basics, or in further applications of capacitive sensing are requested to refer to the comprehensive book from L.P. Baxter (Baxter, 1996), e.g.

In the following, this chapter explains first the basics of capacitance and a capacitor (Section 4.1). Afterward, it introduces capacitive sensing (Section 4.2) and discusses its advantages and disadvantages (Section 4.2.3). The chapter closes with a brief overview of capacitive sensing, specifically in the field of human-computer interaction (Section 4.3).

4.1. Capacitive Sensor Basics

Since capacitive sensing is based on the capacitive coupling principle, this section introduces the capacitor's basics. The presented concepts and principles are simplified, requiring no deeper knowledge in physics, such as Maxwell's equations. L.P. Baxter (Baxter, 1996), e.g., gives a detailed derivation of these equations and concepts.

4.1.1. Electrical Charge

All elements consist of atoms. According to the atom model theory, an atom comprises an atomic nucleus and an atomic shell. The atomic nucleus contains positively charged protons and electrically neutral neutrons. The atomic shell consists of negatively charged electrons moving around the nucleus. An atom is electrically neutral if the number of protons equals the number of electrons. If more electrons

[1] Theremin, invented by Lev Sergeyevich Termen in 1920: https://en.wikipedia.org/wiki/Theremin (accessed on 30.03.2021)

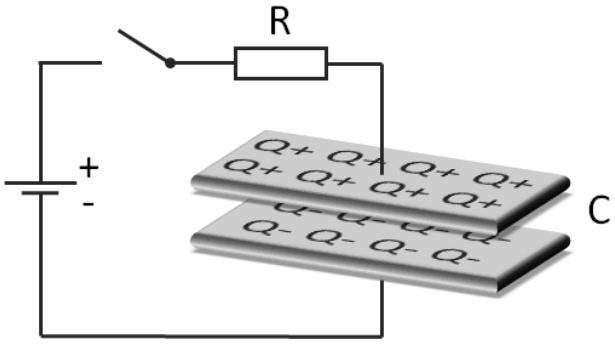


Figure 4.1.: A capacitor storing electrical charge in a circuit.

are present, the atom is negatively charged. If more protons are present, it is positively charged. The electric charge of a body indicates how large its electron surfeit or deficiency is. The charge has the symbol Q, and the unit is C (Coulomb). The electric charge is a multiple of the elementary charge e, where $e = 1.602 \cdot 10^{-19}\mathrm{C}$. Consequently, an object's charge can be expressed as $Q = N \cdot e$, whereas N is the number of charges.

Electric current is the electric charge flow through an object that produces no net loss or gain of electric charge. The interrelation between charge and electric current I can be represented as follows: $Q = I \cdot t$, where t is the time. From this equation, the following relationship is derived: $1\,\mathrm{A\,s} = 1\,\mathrm{C}$. In other words, if a current of one ampere ($1\,\mathrm{A}$) flows through an electrical conductor for $1\,\mathrm{s}$, a charge of $1\,\mathrm{C}$ is transported through the conductor cross-section.

4.1.2. Dielectric Constant

The Coulomb's law describes the force F between two point charges Q_1 and Q_2, separated by r meters: $F = \frac{Q_1 Q_2}{4\pi\varepsilon_0\varepsilon_r r^2}$. The symbol ε represents the permittivity relating to a material's capability to transmit an electric field. The material's net capacitance, thus, depends on the dielectric constant. The constants $\varepsilon_0 = 8.854 \times 10^{-12}\,\mathrm{F\,m^{-1}}$ and ε_r indicate the dielectric constant and the relative permittivity, respectively. The relative permittivity of a material is the dimensionless ratio of its permittivity ε to the permittivity of the vacuum ε_0.

4.1.3. Capacitor

A capacitor is a passive electrical component that stores an electric charge's potential energy in an electric field. Generally, it is composed of two conducting electrodes separated by a non-conducting material. Figure 4.1 depicts a capacitor in a circuit. As soon as the switch is closed, electrical charge is stored on these electrodes. The charge is stored until the potential difference between them is equal

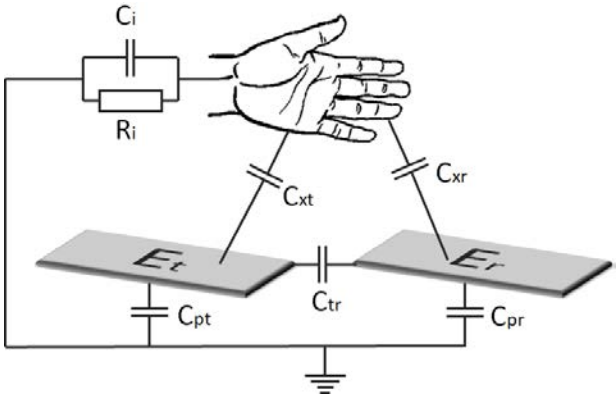


Figure 4.2.: Lumped circuit model.

to the applied source voltage. After disconnecting the source voltage, the electrical charge remains in the electrodes. Since there is no perfect insulator, the charge is lost by leakage. The capacitor also loses charge when another component in the circuit consumes the stored energy.

4.1.4. Capacitance

In simple terms, the electrical capacitance indicates the capacity for electrical charges Q. The symbol of the capacitance is C, and the unit of measurement is Farad F. The electrical capacitance C between two electrically conducting objects separated from each other is equal to the ratio of the stored quantity of charge Q and the electrical voltage U between them: $C = \frac{Q}{U}$. Based on this equation, the capacitance of a plate capacitor can be computed as follows: $C = \varepsilon_0 \varepsilon_r \frac{A}{d}$, where A is the area of each plate in m^2, and d is the distance in meters between the two plates. The detailed derivation is described in (Baxter, 1996), e.g. The capacitance increases with the enlargement of the surface and the plate distance reduction.

4.2. Capacitive Sensing

Capacitive sensing is a technology detecting contactless the presence or absence of any objects. It is based on measuring the change of capacitance of a capacitor system. One of the first studying capacitive sensing in context of human-computer interaction extensively are Smith and Zimmerman (Zimmerman et al., 1995; Smith, 1996, 1999, e.g.) at the MIT. Recent continuing contributions were made at the Technische Universität Darmstadt for example by Wimmer, Braun, and Grosse-Puppendahl (Wimmer et al., 2007b,a; Wimmer, 2011; Braun et al., 2015; Grosse-Puppendahl et al., 2013a,b, 2017, e.g.). In the following, this principle and the measurement modes are explained as well as the advantages and limitations.

4.2.1. Principles of Capacitive Sensing

Capacitive sensing is a technology to measure the change of a capacitor system. Such a capacitor system consists of at least two electrodes in the context of capacitive sensing. Depending on the application, the electrodes can be made of various materials and substances, such as copper, plastic, textiles, paint, or even human skin. As described in section 4.1, a capacitance exists between two electrodes.

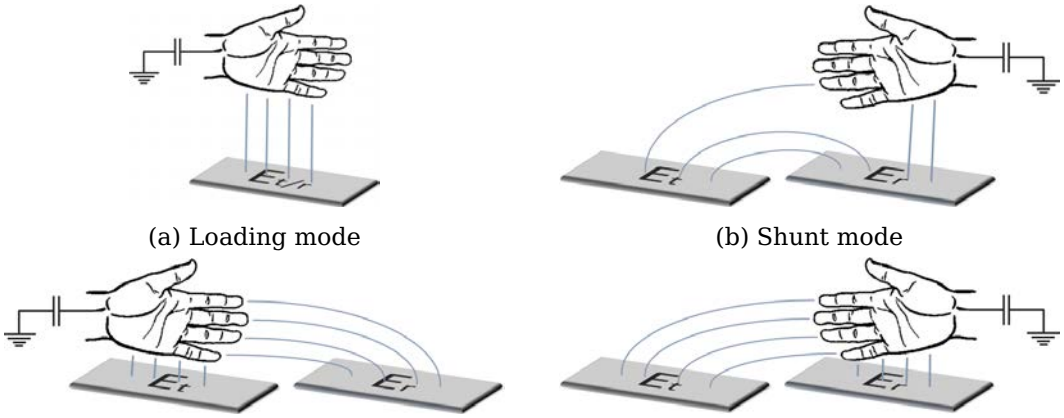
Figure 4.2 depicts a simplified lumped circuit model for a capacitive system with a transmit electrode E_t supplied with a voltage, receive electrode E_r, and an conductive object, in this case a hand. There exists a capacitance C_{tr} between the transmit and receive electrode. If a hand, which is connected to the electrodes via the ground, approaches the electric field between the electrodes, the hand is capacitively coupled into the system. It creates a capacitance C_{xt} and C_{xr} between the electrodes changing the capacitance C_{tr}. All electrodes in the system must be connected through a common ground (Baxter, 1996). In the simplest case, it can be the earth. The capacitances C_{xt} and C_{xr} are proportional to the overlap between the corresponding electrode and the opposed conductive object. The relationship between capacitance C, the area of overlap A, and the object proximity d is given by $C = \varepsilon_0 \varepsilon_r \frac{A}{d}$, where ε_0 is the dielectric constant and ε_r the relative permittivity (see also Section 4.1). In addition to the capacitances C_{xt}, C_{xr}, and C_{tr}, further capacitances are included in the measurement, such as the intrinsic human body capacitance C_i and the inevitable parasitic capacitances C_{pt} and C_{pr}, which are present in every physical circuit. In principle, device pins and the presence of ground return lines in a circuit design add parasitic capacitances that are summed up in the measured total capacitance.

The capacitance at the measuring electrode can be determined in different ways. In the case of a DC voltage, it can be approximated by measuring the time it takes to charge and discharge the electrode. This method is very simple, but it is also very limited in resolution and very slow in measurement, so it is primarily suitable for touch-buttons. Usually, however, an AC voltage is applied, and thus, an oscillating field is generated. The capacitance is then usually determined by the change of the displacement current.

Many factors influence the measurement, such as temperature variations, circuit crosstalk, improper grounding, or shielding. All factors that must be considered in circuit design are described in detail in (Baxter, 1996).

4.2.2. Measurement Modes

In the application domain of human-computer interaction, there are different sensor configurations for measuring capacitance (Grosse-Puppendahl et al., 2017). First, it is distinguished between active and passive capacitive sensing. In active sensing, the transmit electrode E_t is controlled driven by a specific voltage. The transmit



(a) Loading mode (b) Shunt mode

(c) Transmit mode (d) Receive mode

Figure 4.3.: Capacitive measuring modes.

electrode E_t is capacitive coupled with the receive electrode E_r. Passive setups in the opposite are driven by environmental sources emitting an electric field.

The measurement modes are classified by the relative position between the human body and the transmit and receive electrodes (E_t and E_r respectively). Figure 4.3 depict simplified these modes for active capacitive sensing.

The simplest mode is the loading mode (Figure 4.3a). A single electrode is driven with a specific voltage creating an electric field. Any object sharing the same ground increases the system's capacitance by approaching the electrode.

The shunt mode consists of a transmit and receive electrode (Figure 4.3b). The transmit electrode generates an electric field between both electrodes. Approaches a conductive object the electric field, it is coupled to the electrodes and grounds of the electric field. This reduces the displacement current between the transmit and receive electrodes.

The transmit mode also uses a transmit and a receiving electrode generating an electric field between them (Figure 4.3c). The difference to the shunt mode is that the human body is very close to the transmitting electrode. The body acts as an extension of the transmit electrode, increasing the range between the body and the receiver electrode. The receive mode is the inverse of the transmit mode (Figure 4.3d). Now, the body is close to the receive electrode.

4.2.3. Advantages and Limitations of Capacitive Sensing

Capacitive sensing has a couple of advantages in opposite to other sensing technologies. The most significant advantage is probably the variety of materials for the electrodes. The electrodes can have any size and shape. They can have the dimensions of square meters or tiny square millimeters. They can be curved, flexible, or stretchable. Equally diverse are the material options, which classically range from

copper over textiles to paint. This allows for the unobtrusive and diverse placement of the electrodes. In addition, the electrodes can be hidden under other materials, such as the desk surface, without affecting the measurement. This also means no occlusion problem as known from other sensors, such as cameras or ultrasonic sensors.

The electronics also usually use low-cost and low-power components. Another advantage of capacitive sensing is that data rates of $100\,\mathrm{Hz}$ and more can easily be achieved. In addition, the data has low complexity and can therefore already be pre-processed and analyzed on the microcontroller.

The electrodes have a wide field of view at a very close distance. On the other hand, capacitive sensors can not focus because of their one-dimensionality. It can only be measured that the object is close to the electrode but not in which area. Another potential difficulty is imposed by the nonlinear relationship between proximity and measured capacity. While movements close to the electrode result in drastic changes in capacitive measurements, its sensitivity decreases at increasing ranges quickly.

4.3. Applications in the Context of Human-Computer Interaction

Capacitive sensing is a well-known technology since many decades. It has already been widely used for a long time, especially in the industrial environment, for example for measurement of pressure, liquid level, balances, linear position, thickness of objects, and proximity or as sensor technology for electronic switches and sliders, accelerometer, digital level sensors, proximity sensors, microphones, vehicle detection sensors, pressure sensors, touch screens, or track pads. However, research and applications in the field of human-computer interaction hardly existed for many years. For a long time, applications were limited to capacitive touchscreens, buttons, and sliders, with a few exceptions. In the nineties, Smith and Zimmerman (Zimmerman et al., 1995; Smith, 1996, 1999, e.g.) at MIT started to investigate capacitive sensing in the context of human-computer interaction and to create applications with it. In recent years, the advantages of capacitive sensing have been increasingly recognized, and hundreds of diverse applications have been implemented with it. Grosse-Puppendahl et al. (Grosse-Puppendahl et al., 2017) have done a comprehensive survey classifying the approaches using capacitive sensing in the field of human-computer interaction. In the following, a selection of works that show the diversity of applications is presented.

The application cases are very diverse. They include, for example, indoor localization (Valtonen et al., 2012; Sousa et al., 2013, e.g.), user distinction (Pickering, 2008; Dietz and Leigh, 2001, e.g.), gesture recognition (Matthies et al., 2021; Wimmer, 2011; Grosse-Puppendahl et al., 2014, e.g.), interaction with everyday objects

(Sato et al., 2012; Zhang et al., 2017c, e.g.), on-body and on-skin interaction (Zhang et al., 2016; Lissermann et al., 2013, e.g.), around-device interaction (Zhou et al., 2016, e.g.), and multi-touch displays (Voelker et al., 2015; Lee et al., 1985, e.g.).

Popular applications also include smart textiles, which are equipped with capacitive sensors to enable, for example, touch interaction (Singh et al., 2015; Poupyrev et al., 2016, e.g.), gesture recognition (Bello et al., 2021; Gong et al., 2021; Wu et al., 2018b, e. g.), activity recognition (Wimmer et al., 2007b; Cheng et al., 2013; Rantanen et al., 2013, e.g.), or vital data tracking (Ueno et al., 2007; Oum et al., 2008, e.g.).

Capacitive sensing finds also application in the context of wearable-based finger and micro gesture recognition. Nguyen et al. (Nguyen et al., 2019) developed, for example, a data glove using electrodes on the fingertips enabling micro gesture interaction. Bian et. al(Bian and Lukowicz, 2021), GestureWrist (Rekimoto, 2001), and CapBand (Truong et al., 2018) apply capacitive sensing to measure arm contour changes caused by finger movements. Kao et al. (Kao et al., 2015) uses capacitive touch sensors on a nail cover for micro gesture interaction too. Boldu et al. (Boldu et al., 2018b) equipped a ring with a capacitive touch area to enable sliding gestures performed with the thumb on the ring worn at the index finger. ElectroRing (Kienzle et al., 2021) utilizes capacitive sensing to recognize pinch gestures and touch events on objects.

5. PeriSense

The previous part of this work pointed out the advantages of rings as interaction devices. It also discussed the limits of current rings and the challenges to be addressed. A limitation of the current rings is the lack of the possibility to enable multi-finger interactions. This limits the interaction space and, thus, the application. This dissertation evaluates the concept of using electric fields for multi-finger interaction with a ring. This chapter presents the conceptual thoughts of using electric fields for multi-finger interaction (Section 5.1). Afterward, this chapter describes the prototypical implementation of the proposed concept (Section 5.2). To distinguish from previous and later ring versions, this prototype generation is called PeriSense.

5.1. Concept

Driven by the attractiveness of rings as an interaction device, but with the limited possibility of recognizing multi-finger gestures, I came up with the basic idea that electrical fields could be applied around the ring. The idea here is that the fingers influence and change the fields differently depending on their position (Figure 5.1).

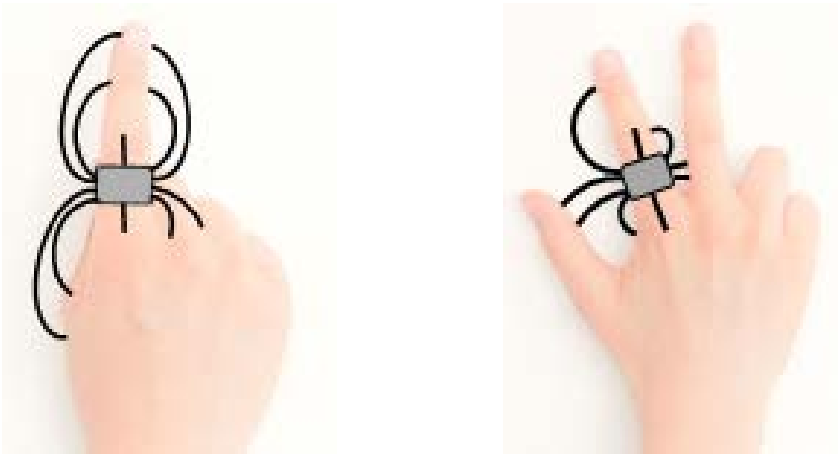


Figure 5.1.: Visualization of the basic idea: The ring creates electric fields around the ring. Depending on the finger pose, the fields are different influenced.

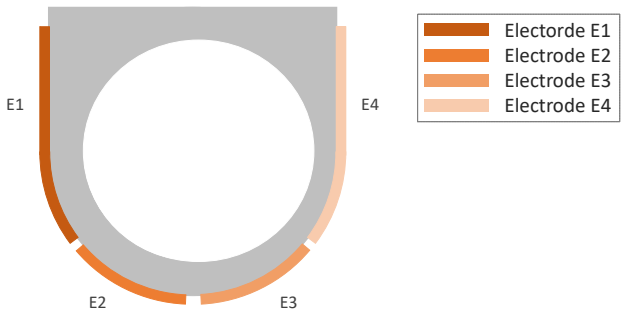


Figure 5.2.: Electrode arrangement around the ring.

These field changes are measurable. Initial tests and research quickly led to capacitive proximity sensing. Through capacitive sensing, changes in the electric field can be detected. These changes are determined by changes in the capacitance of the capacitive sensor.

The concept is based on arranging capacitive electrodes on a ring and employing capacitive proximity sensing to quantify the distances to the adjacent fingers. Measuring proximity between conductive objects can be achieved effectively by capacitive sensing in several different modes (see Section 4.2.2). These measurement methods are the loading, shunt, receive, and transmit mode. In shunt mode, it is more difficult to measure the influence of conductive objects at a distance, especially when the electrodes are placed close together around the ring. Two electrodes per field are required, whereas, in loading mode, only one electrode per field is needed. Besides, when arranging the electrodes, care must be taken to ensure that the fields do not influence each other. The transmit and receive modes are not suitable for this scenario, as they makes use of the electromagnetic emissions of the environment. This is difficult in everyday situations because the ambient electromagnetic emissions are different in each environment. It is also questionable whether finger movements can be detected at all, at least without special calibration or environmental setup. We apply the loading mode, its purest form, which drives a single electrode with an oscillating signal, leading to the periodic charging and discharging of the respective electrode. Such an electrode acts as one plate of a capacitor while the surrounding conductive objects in its environment act as the opposite plate connected to ground. The capacitance itself is proportional to the size of the electrode and inversely proportional to the displacement to its complement. By approaching the electrode with such a conductive object, its charge time increases consequently as there is more energy to store at a higher capacity. The capacitance can thus accordingly be computed by measuring those changes in frequency. As in such a scenario, the human body serves as the ground electrode;

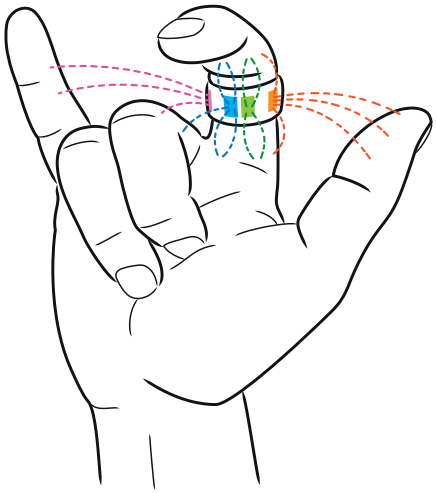


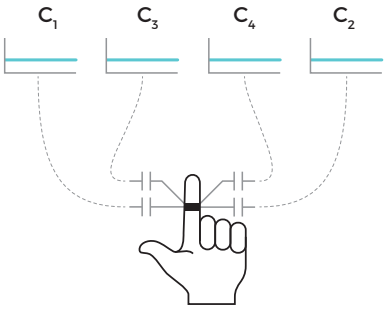
Figure 5.3.: Simplified conceptual visualization of the electrode placement and the flow of the field lines.

the human body needs to share the same ground potential as the electronic sensing device.

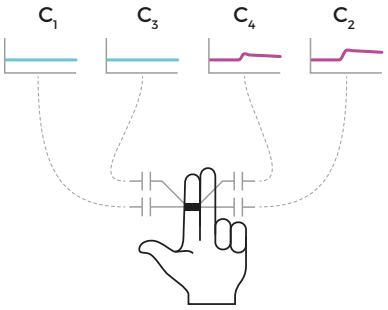
The implementation concept provides that four electrodes are placed around the ring (Figure 5.2). The four electrodes are arranged in a linear array on the ring to enable a form of spatial perception. Two of the four electrodes are directed sidewards to measure the influence of the neighboring finger primarily. If the ring is worn on the index finger, then the distances to the thumb and little finger are considerable. Therefore, the electrodes are larger on the sides. The other two electrodes are directed towards the palm and gather proximity data below the wearing finger to detect finger bends, also from the neighboring fingers partially (Figure 5.3).

Figures 5.4a to 5.4c illustrate the working principles symbolically. If the only outstretched finger is the index finger (Figure 5.4a), then the capacitive sensor values are rather low. Approaching the ring with the middle finger (Figure 5.4b) or the thumb (Figure 5.4c), the particular capacitive sensor measurement values increases.

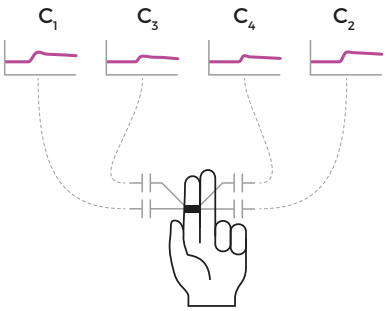
In contrast to a lot of other sensing modalities, capacitive sensing can sense a wide field of view at close distances without the need for a lens (Grosse-Puppendahl et al., 2017). The downside is that it is not trivially possible to differentiate the causes of changes in total capacitance for a single electrode. In order to overcome this limitation, an active shield is driven at the same potential as the electrode under measurement. Although the overall sensitivity is reduced, the measurement can be blocked partially and directed in particular directions (Wang, 2015).



(a) Pointing with index finger.



(b) Outstretched middle finger.



(c) Move thumb close to index finger.

Figure 5.4.: Capacitive proximity sensing working principles.

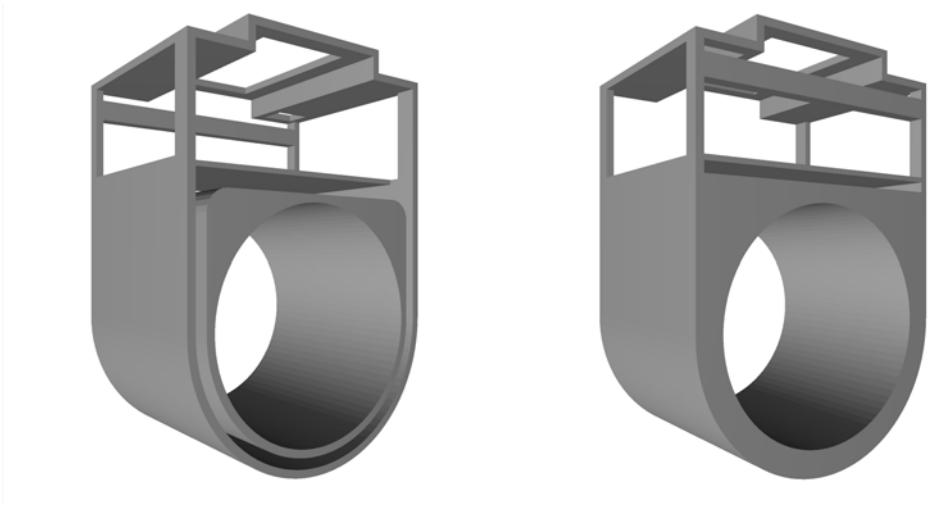


Figure 5.5.: 3D model of the PeriSense case.

5.2. Implementation

In this section, the implementation of the concept is described. First, the hardware implementation of the prototype is described (Section 5.2.1). Afterward, the software implementation is described, starting with the firmware (Section 5.2.2), and finally the driver (Section 5.2.3).

5.2.1. PeriSense Hardware Prototype

From the idea to the prototype presented here, there were many prototype versions. With each new version new insights were gained and taken into account in a further improved prototype. The history of the prototypes is briefly explained and described in the appendix (Appendix A).

Short traces and a reproducible design are crucial requirements. Therefore, a layout for a 2-layer flexible circuit board was designed in Eagle[1]. This way, the copper electrodes for capacitive proximity sensing can be laid out in an editor for reliable production by a common PCB manufacturer. The copper electrodes are designed as rectangles with rounded corners, with lengths of edges ranging from 12 to $17\,\mathrm{mm}$, forming areas from approximately 200 to nearly $280\,\mathrm{mm}^2$. The electrodes pointing downwards are smaller than the electrodes pointing side-wards. Although the measurable distance decreases with smaller electrodes, this limitation is mitigated as these two electrodes primarily detect finger bending. The two electrodes on the side, however, primarily measure the distance to the adjacent fingers. This distance

[1] EAGLE is an electronic design automation software for the automation of electronic design: https://www.autodesk.de/products/eagle/overview (accessed on 08.09.2020)

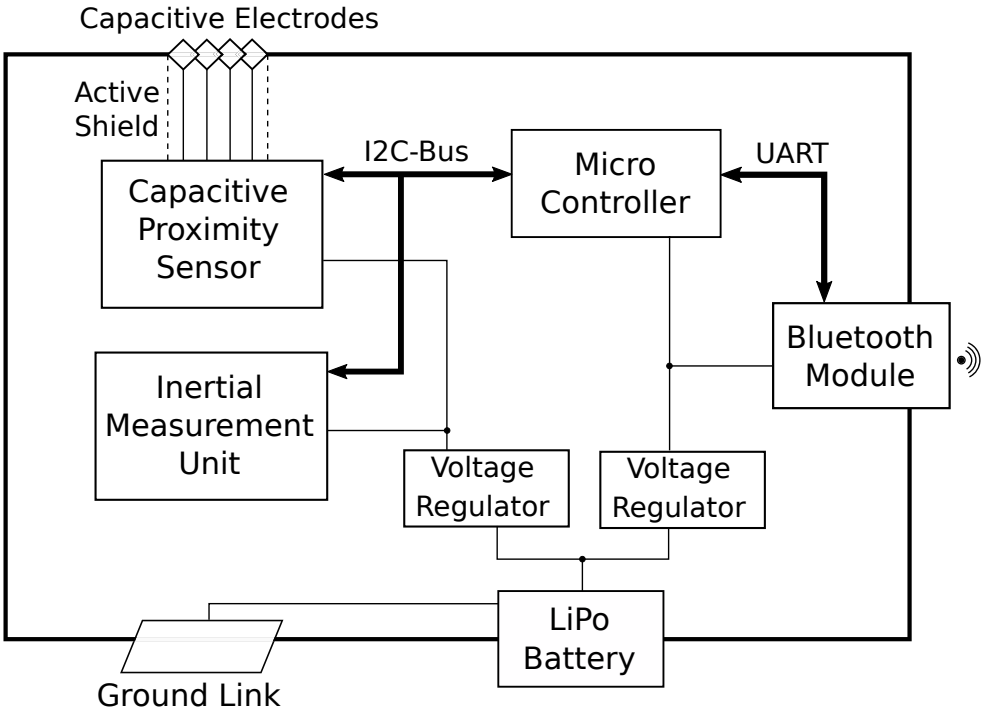


Figure 5.6.: System architecture block diagram.

can measure a few centimeters and is the reason why these electrodes are laid out larger.

The thickness of a copper layer is specified by $35\,\mu\mathrm{m}$. The active shield is laid out below the electrode layer to direct the capacitive measurements outwards. The flexible base material allows for a convenient way of placing all electrical components in a 3D-printed finger ring casing (Figure 5.5). The 3D model was created with FreeCAD[2]. The dimensions of the ring case are a width of $22\,\mathrm{mm}$, a height of $44\,\mathrm{mm}$, a variable inner diameter between 18 and $22\,\mathrm{mm}$ (depending on the finger size and prototype), and outer diameter of $27\,\mathrm{mm}$. The cases of our prototypes have a thickness of $2.5\,\mathrm{mm}$ in this area, with a shell thickness of $0.5\,\mathrm{mm}$.

In order to reduce cross-talk between sensor measurements and their processing and transmitting, the flex-board is segmented into two parts with dedicated voltage regulators. For the capacitive sensing measurement, a Texas Instruments FDC1004 sensor (TexasInstruments, 2015) is used. It features a 4-channel capacitance-to-digital converter for capacitive proximity sensing. Each channel is connected to one of the four electrodes which are sampled in sequential order. As the ring device

[2]FreeCAD is an open-source parametric 3D modeler: https://www.freecadweb.org/ (accessed on 20.08.2020).

(a) Top side. (b) Bottom side.

Figure 5.7.: Assembled flexible circuit board.

and the human body have to share the same ground potential for this sensing mode to function correctly, the inside of the ring is coated with a copper foil to provide an electrical connection to the worn finger. An additional inertial measurement unit (IMU) with 9 degrees of freedom is equipped to augment sensor readings for gesture recognition with hand position and orientation tracking. For the IMU, a InvenSense MPU9250 sensor (InvenSense, 2017) was applied. Figure 5.6 provides a block diagram for illustration of the underlying system architecture. The circuit layouts and complete component list can be found in the appendix B.

The flexible circuit boards were produced by a dedicated service provider and assembled using a reflow soldering technique. Figure 5.7 shows an assembled circuit

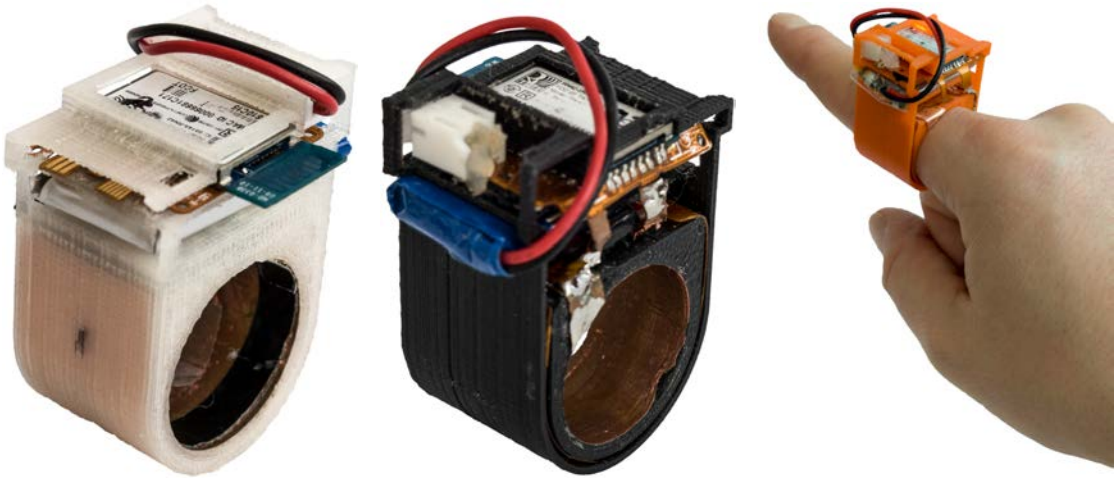


Figure 5.8.: PeriSense prototypes in different colors and sizes.

Table 5.1.: Description of the LED status.

LED color	State	Description
Red	On (solid)/Off	Power is on/off.
Orange	Blinking	Bluetooth module is in discovery mode.
	Solid	There is a established Bluetooth connection.
Green	Blinking	The ring sends data via the Bluetooth interface.

board. The top segment includes a Microchip ATmega328P[3] for reading out the respective sensor measurements sequentially using an I2C bus and transmitting those readings with the help of a Bluetooth 2.1 module connected over UART at a baud rate of 115200bps. The device is driven by a small-sized LiPo battery with a capacity of $110\,\mathrm{mA\,h}$. Consuming about $128\,\mathrm{mA}$ at $3.0\,\mathrm{V}$ on average during measurement and Bluetooth communication, a battery charge lasts for around an hour of continuous usage. By disabling Bluetooth communication and using UART exclusively, average power consumption went down to $16\,\mathrm{mA}$. Without any energy optimizations the average run-time is about 1 to 1.5 hours. Three LEDs sign the devices status (Table 5.1). Figure 5.8 shows the complete assembled rings in different colors and sizes.

Figure 5.9 shows a record of capacitive raw values of electrode E1 to E4 while the ring was worn on the index finger. Each electrode has a different offset. The offset can change with the body's internal resistance (which varies e.g., with the body's fluid balance) and can be influenced by other parasite capacitances. As closer a finger part approaches the electrodes as larger is the sensor raw value. In areas 1 and 4, the index finger was stretched, and all other fingers bend so that no finger part was in the range of the electrodes. In area 2, all fingers are stretched out. Consequently, electrodes E1 and E4 are touched by the thumb and middle finger. In area 3, all electrodes have contacts with finger parts because the fingers formed a fist. In area 5, electrode E1 and E2 measure the effect of the thumb touching the ring while the index finger is stretched out. In area 6, the fingers were randomly moved.

5.2.2. Firmware

The ring firmware is implemented in C++. To read sensor data from the I2C bus, the I2Cdev library[4] is used. This library provides already a module to configure and read the MPU9250. The I2Cdev library was extended by a self-implemented module for the TI FDC10004 sensor. Both sensors are read as fast as possible in polling mode. Every time new data is available, it is send over the Bluetooth serial interface. This results in a transmission frequency of about $100\,\mathrm{Hz}$. There is no further

[3] Microchip ATmega328P documentation: https://www.microchip.com/wwwproducts/en/ATmega328p (accessed on 20.08.2020)

[4] I2Cdevlib written by Jeff Rowberg: https://github.com/jrowberg/i2cdevlib

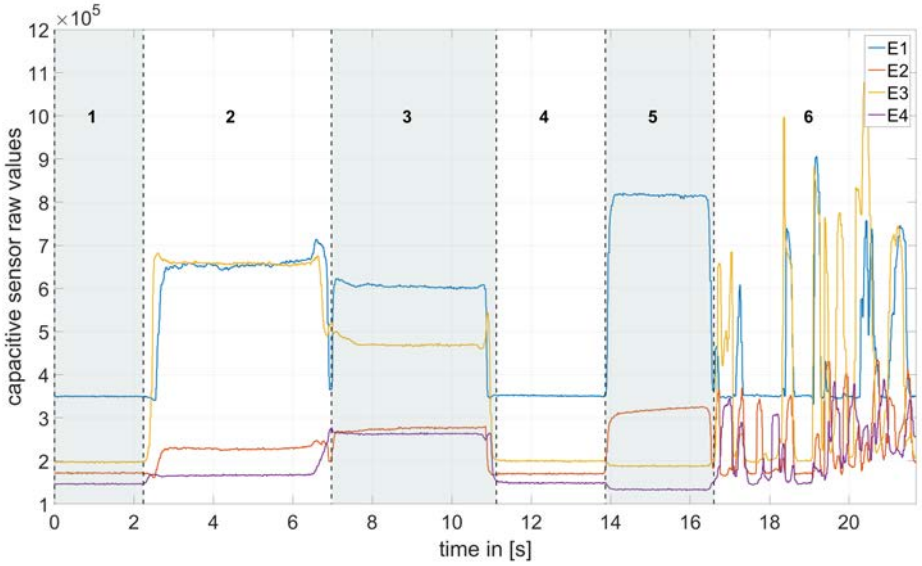


Figure 5.9.: Example of capacitive raw values of electrode E1 to E4 while the ring was worn on the index finger.

data processing by the firmware, because of the limited resources of the microcontroller. The memory size for the EEPROM and RAM of Microchip ATmega328P microprocessor, however, are only $1\,\mathrm{kB}$ and $2\,\mathrm{kB}$, respectively. Thus, the resources of the microcontroller are almost entirely utilized. The implementation of additional features bears the risk of affecting the microcontroller's performance, and thus the transmission frequency of $100\,\mathrm{Hz}$ could not be guaranteed.

After power on, the firmware initializes the FDC1004 and the MPU9250 sensors. The Bluetooth module is activated and set to discover mode. This allows the ring to be found by other Bluetooth devices like smartphones or laptops. If a device is paired with the ring, the ring starts sending the sensor data via the serial Bluetooth interface. At every time point t, a 17-dimensional data vector is transmitted. Table 5.2 describes the data the ring sends at every time step t. To maximize the transmission rate, the data is transmitted as bytes. To ensure that the byte sequences are correctly separated and converted to the corresponding data types, each byte vector is encoded with the Consistent Overhead Byte Stuffing (COBS) algorithm (Cheshire and Baker, 1997). The COBS algorithm uses the zero as a separator or end signal of the byte vector. Consequently, all zeros are removed from the byte vector and replaced by pointers pointing to the next zero. A so-called overhead byte at the beginning of the byte packet points to the first zero. Thus 43 bytes every time step t are transferred, 41 bytes of them are data and the 2 remaining bytes from the COBS encoding.

Table 5.2.: Description of the transmitted ring data.

Id	Size	Type	Description
ts	4 B	Int32	Time stamp in milliseconds starting from 0 at ring startup
di	1 B	Int8	A legacy data field used for debugging purpose in the development phase (is deprecated).
e1	4 B	Int32	Capacitive raw measurement of electrode 1
e2	4 B	Int32	Capacitive raw measurement of electrode 2
e3	4 B	Int32	Capacitive raw measurement of electrode 3
e4	4 B	Int32	Capacitive raw measurement of electrode 4
ax	4 B	Int16	Acceleration raw measurement along x-axis of the motion sensor
ay	2 B	Int16	Acceleration raw measurement along y-axis of the motion sensor
az	2 B	Int16	Acceleration raw measurement along z-axis of the motion sensor
gx	2 B	Int16	Raw measurement of the angular velocity around x-axis of the motion sensor
gy	2 B	Int16	Raw measurement of the angular velocity around y-axis of the motion sensor
gz	2 B	Int16	Raw measurement of the angular velocity around z-axis of the motion sensor
mx	2 B	Int16	Raw measurement of the magnetization along x-axis of the motion sensor
my	2 B	Int16	Raw measurement of the magnetization along y-axis of the motion sensor
mz	2 B	Int16	Raw measurement of the magnetization along z-axis of the motion sensor
tmp	2 B	Int16	Temperature raw measurement of the motion sensor

5.2.3. Driver

Due to the use of Bluetooth for data transmission, the ring can be connected in many ways. The ring is connected via the serial Bluetooth interface RFCOMM. Many devices and operating systems support this technology. Accordingly, most programming and script languages offer platform-independent interfaces. Drivers were developed in Python and, in the beginning, also experimental drivers in Java and Matlab. Python was chosen because it allows data analysis, and the results can be implemented directly in the application without the need for a new implementation.

In the driver, the COBS packets are decoded, and the resulting byte vector is converted into the raw data, as shown in Table 5.2. The raw data from the IMU is converted into the corresponding physical values according to the sensor documentation (InvenSense, 2017). The capacitive raw values are not converted because it is only a linear conversion factor and is irrelevant for further considerations. Table 5.3 shows the data provided by the driver.

The driver was implemented as a Python class. The most essential functions are:

```
class PeriSense:
        def __init__(self, serial_port)
        def connect(self)
        def disconnect(self)
        def addHandler(self, handler)
        ...
```

The constructor expects the serial interface port as a string value to which PeriSense is connected. On Windows, for example, the string is "*COM4*" or on Linux, for example, "*/dev/rfcomm0*". The operating system assigns the corresponding port number. The serial interface is opened by invoking `connect()`, and thus the data transfer is started. The data transfer and connection to the ring can be disconnected using `disconnect()`. The driver provides the received ring data via an Observer pattern. For this purpose, a handler, i.e., a function with the following signature, must be passed to the driver:

```
def handler(ts, di, e1, e2, e3, e4, ax, ay, az, gx, gy, gz, mx, my, mz
    , tmp)
```

The function parameters correspond to the values in table 5.3.

Table 5.3.: Description of the data provided by the driver interface for further processing.

Id	Type	Data Range	Unit	Description
ts	Int32	0 to 21474836	ms	Time stamp in milliseconds starting from 0 at ring startup
di	Int8	-127 to 128	-	A legacy data field used for debugging purpose in the development phase (is deprecated).
e1	Int32	0 to 21474836	-	Capacitive raw measurement of electrode 1
e2	Int32	0 to 21474836	-	Capacitive raw measurement of electrode 2
e3	Int32	0 to 21474836	-	Capacitive raw measurement of electrode 3
e4	Int32	0 to 21474836	-	Capacitive raw measurement of electrode 4
ax	Double	-4 to 4	g	Acceleration in x-direction
ay	Double	-4 to 4	g	Acceleration in y-direction
az	Double	-4 to 4	g	Acceleration in z-direction
gx	Double	-1000 to 1000	$^\circ/s$	Angular velocity around x-axis
gy	Double	-1000 to 1000	$^\circ/s$	Angular velocity around y-axis
gz	Double	-1000 to 1000	$^\circ/s$	Angular velocity around z-axis
mx	Double	-4800 to 4800	μT	Magnetization in x-direction
my	Double	-4800 to 4800	μT	Magnetization in y-direction
mz	Double	-4800 to 4800	μT	Magnetization in z-direction
tmp	Double	-40 to 85	$^\circ C$	Temperature

6. Evaluation

The previous chapter introduced the concept of tracking multiple fingers' movement with one ring using capacitive sensing. Additionally, the implementation of PeriSense, a prototype that implements the idea prototypical, was described. This chapter presents a comprehensive evaluation using the PeriSense prototype. The evaluations presented in this chapter are analytical and not bound to applications and aim to determine the interaction space. Consequently, the used gesture sets do not necessarily relate to real-world applications. The determination of the interaction space should allow interaction designers to create their specific interaction applications. Some examples of implementations and evaluations in real-world application-context are presented in chapter 7.

In the following, the evaluation of the technical properties such as the influence of environmental noise and the spatial resolution of the electrodes (section 6.1), the static interaction space determining which fingers are in the range of the measurement (section 6.2), the dynamic interaction space determining which finger movements are detectable (section 6.3), the around-device interaction space showing which finger movements performed around the ring (section 6.4), and, finally, finger tracking (section 6.5) are presented.

6.1. Technical Evaluation

To determine the sensor properties of PeriSense, two experiments were conducted. First, the influence of external sources and noise were studied. Second, the spatial resolution of the four electrodes was evaluated.

6.1.1. Experiment I: Influence of Ambient Electrostatics

Since we are surrounded by natural and artificial electrostatic noise sources such as power lines or electrical appliances, it is to study if these sources influence capacitive sensing performance first. For this purpose, data at eight different locations were recorded. The ring was placed in the respective locations and not worn on a finger during this experiment to exclude signals caused by unavoidable finger and hand movements. To avoid disturbances due to battery changes, the original battery was replaced by a 18650 battery with $3.5\,\mathrm{A\,h}$, which was placed next to the ring.

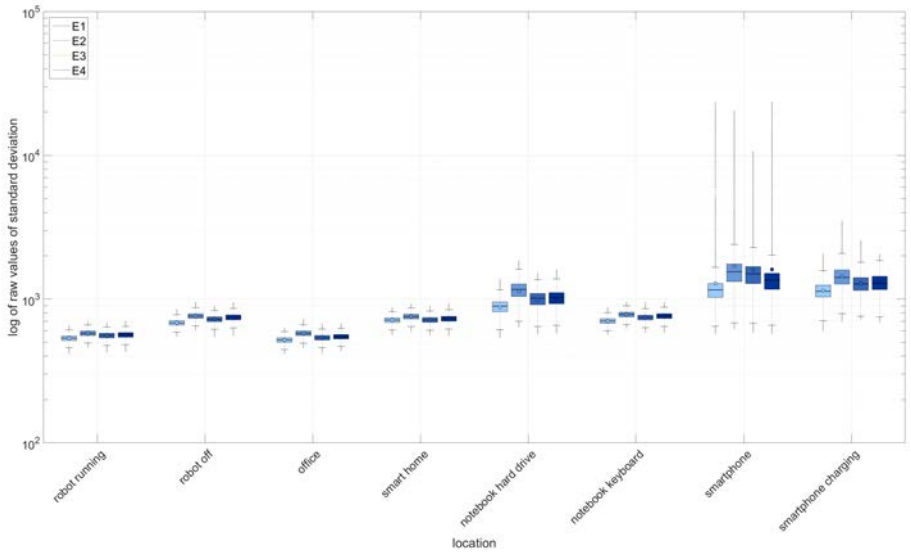


Figure 6.1.: Distributions of the standard deviations of five seconds long sliding window over 4.5 hours lasting measurements at different locations.

The following locations were evaluated: close to a big robot[1] which was running, close to a big robot which was turned off, on a table in an office, in a smart home environment at a$10\,\mathrm{cm}$ distance from a 98" running TV, on a running notebook above the hard drive (SSD) area, on a running notebook above the keyboard, on the backside of a non-charging smartphone and on the backside of a charging smartphone in flight mode.

For each location, 4.5 hours of data were recorded. On each data set, a sliding window with a size of five seconds was applied. The window was moved with a step size of one. For each window, the standard deviation and mean value were calculated. Figure 6.1 shows the boxplots of the standard deviations at each location for the four electrodes E1 to E4. Since some boxplots have large outliers, a logarithmic scale was applied. There is no noticeable difference in the distribution of the standard deviation at the robot (regardless of running or not), in the office, at the TV in the smart home, and on the notebook's keyboard. On the smartphone's backside, the standard deviation distribution is similar to the other locations, such as the laptop or smart home. However, there are periodically occurring peaks in the measurements. These result from communication modules such as mobile, wireless network or Bluetooth and cause huge outliers and a little broader distribution. A separate test showed that they do not occur when the phone is in flight mode. A significantly broader distribution of the standard deviation can be observed over the notebook's hard drive and on the charging smartphone's backside in flight mode.

[1] Care-O-Bot 3 robot: https://www.care-o-bot.de/en/care-o-bot-3.html (accessed on 02.09.2020)

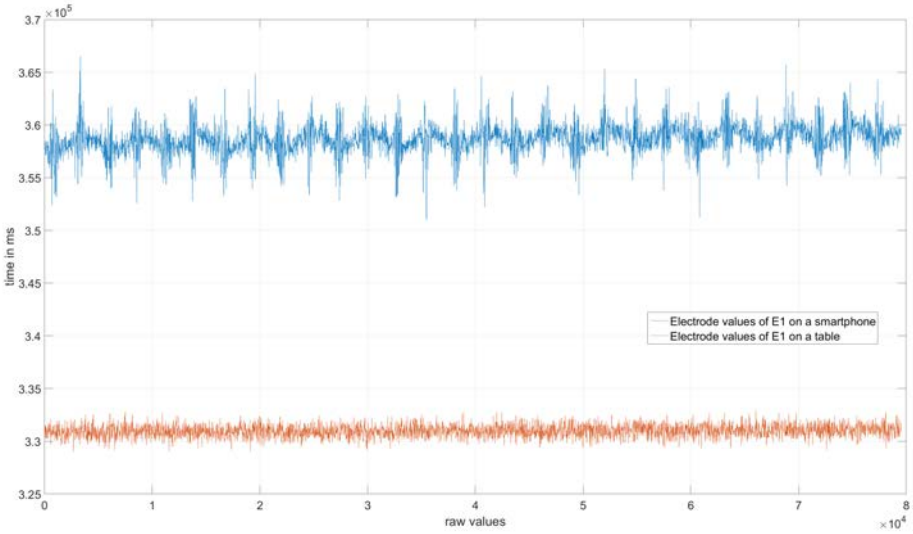


Figure 6.2.: A sample for measurement for electrode E1 in the office and above the loading smartphone in flight mode.

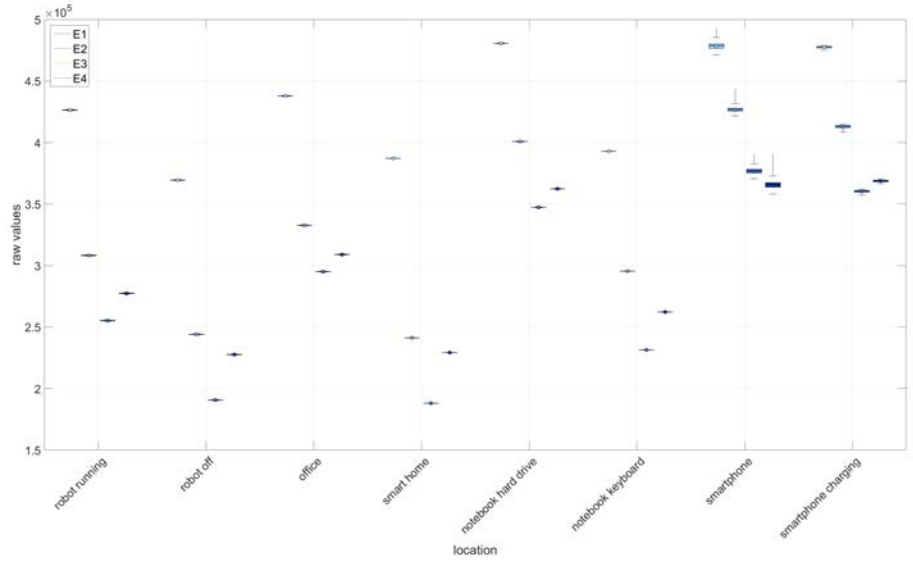


Figure 6.3.: Distributions of the mean values of five seconds long sliding window over 4.5 hours lasting measurements at different locations.

Figure 6.4.: Experimental set-up for evaluation of spatial sensor resolution.

The laptop's SSD and the charging smartphone induce an oscillating noise. Figure 6.2 shows a section for the measurement for electrode E1 in the office and above the charging smartphone.

The mean value variance is small for each location (Figure 6.3), and no signal drift over time at constant temperatures can be observed. The slightly broader mean distribution for the smartphone results from the high noise. In summary, it can be concluded that ambient electrostatics seems not to be a problem for PeriSense. Only some devices in limited situations can induce noise. However, it is to consider that these measurements were performed without wearing the ring on the finger. That means the was no reference ground. Wearing the ring on the finger, the ring and the human body share the ground stabilizing the measurements and reducing the noise. Additionally, these distortions occur only with direct contact and disappear with some distance of centimeter or even millimeter. Consequently, it is to conclude that the measured disturbances cause no issues in real-life applications.

6.1.2. Experiment II: Spatial Resolution

In order to determine the effective interaction space of PeriSense, the spatial resolution for each electrode was computed by measuring the capacitive proximity in

a grid surrounding the ring. The spatial resolution is defined by Lucas et al. as "the smallest distance between two identical objects that produce a signal with a measurable difference compared to the signal they would produce if they were superimposed"(Lucas et al., 2015). The computation of the spatial resolution was based on the procedure for flat electrodes employed in (Grosse-Puppendahl et al., 2013a). Therein, the spatial resolution is determined by computing the mean $\mu(d)$ and standard deviance $\sigma(d)$ across measurements of the same true proximity d and measuring the maximum pairwise distance between the border measurements that falls into the range defined by $(\mu(d) - \sigma(d), \mu(d) + \sigma(d))$.

Opposed to the usage of flat electrodes, in this case, the electrodes are bent on a ring. This is the reason why measurements were gathered in a grid and not just at one-dimensional distances. For each of these 2D-measurement positions p, a set P_p of all samples at this position was constituted, and the mean $\mu(P_p)$ and the standard deviance $\sigma(P_p)$ were calculated. Subsequently, a set of all measurement positions $P_{p,s}$ was created which fell into the measurement range defined by $(\mu(P_p) - \sigma(P_p), \mu(P_p) + \sigma(P_p))$. In this resulting set $P_{p,s}$, the maximum pairwise difference of distances to the ring-border was taken. This resulted in a one-dimensional distance projection, and therefore is analogous to the method in (Grosse-Puppendahl et al., 2013a).

For executing such an experiment, the limb of a Care-O-Bot 3 robot[2] was equipped with an acrylic bar of $30\,\mathrm{cm}$ to keep the robot's capacitive influence as small as possible (Figure 6.4). At the end of the bar, an aluminum tube with a length of $10\,\mathrm{cm}$ and a diameter of $2\,\mathrm{cm}$ was mounted, simulating a neighboring human finger, similarly to the simulated arm in (Grosse-Puppendahl et al., 2013a). As the simulated finger and PeriSense have to share the same ground potential for optimal performance, the aluminum tube was connected to the exposed ground link via a small diameter wire to achieve comparable results. The ring itself was put on a 3D printed mount.

The robot traversed the measurement grid from left to right, top to bottom in steps of $5\,\mathrm{mm}$. All positions refer to the center of the ring and represent the center of the aluminum probe. As the probe has a diameter of $2\,\mathrm{cm}$, this accounts for the minimum measurement gap of $1\,\mathrm{cm}$ around the ring in the resolution plots. At each measurement position, $5\,\mathrm{s}$ of capacitive values were read after a short settling time of $2\,\mathrm{s}$. This procedure lasted about 5 hours, and thus exceeded the small battery capacity of the prototype. As before, the small battery was replaced with a bigger 18650 type battery with $3.5\,\mathrm{A\,h}$, which was put aside on the table.

Figure 6.5 contains a plot of the computed resolution values of each electrode for each measurement position. At low proximity, the electrodes have a resolution in millimeter range. The figure also depicts which areas are exclusively measured by a single electrode and which areas are overlapping. Up to a displacement of approximately $5\,\mathrm{cm}$ most of the surrounding positions are sensed by at least two

[2] Care-O-Bot 3 robot: https://www.care-o-bot.de/en/care-o-bot-3.html (accessed on 02.09.2020)

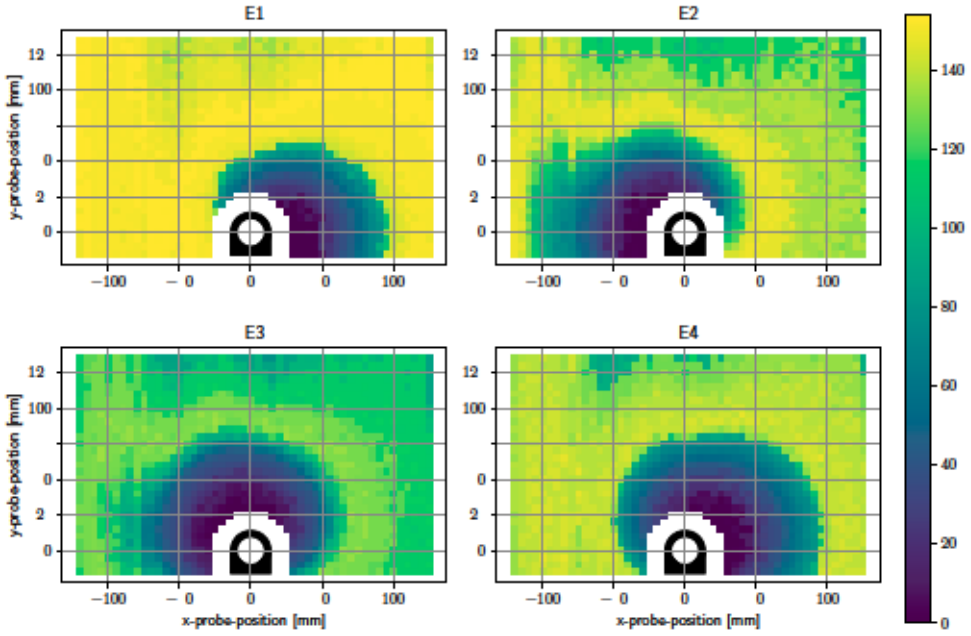


Figure 6.5.: Spatial resolution for each electrode. The color represents the particular resolution in millimeters.

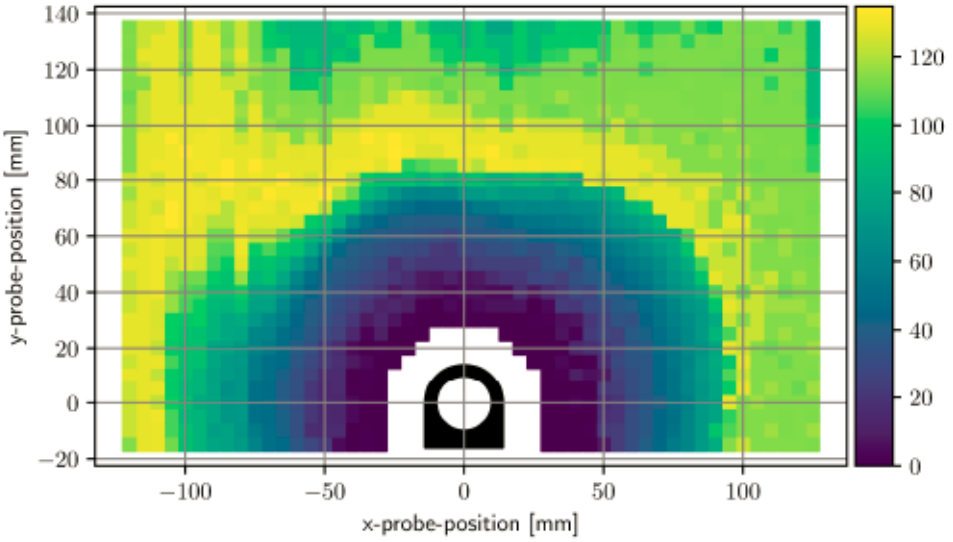


Figure 6.6.: Superimposed plot of spatial resolutions from all of the four electrodes.

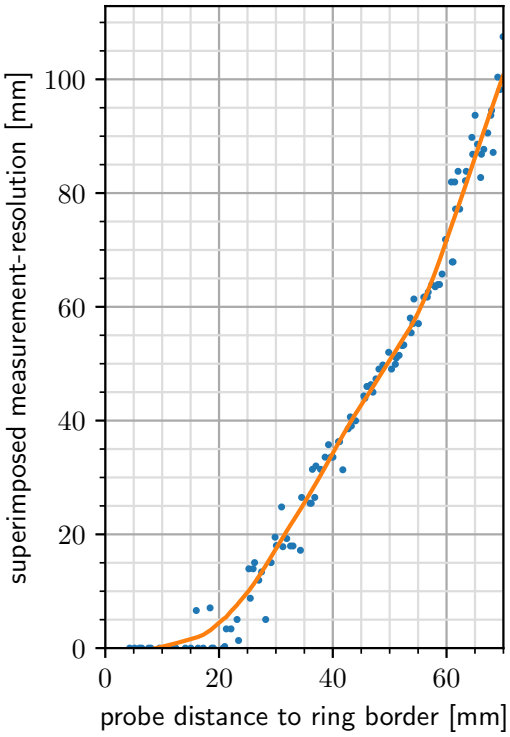


Figure 6.7.: Spatial resolution at distances measured between ring-border and tube-border.

electrodes. Figure 6.6 shows a superimposed plot of all four electrodes by taking the minimum of each resolution. Figure 6.7 depicts the superimposed resolution in relation to the actual distance between ring border and probe border. It suggests that up to $1.5\,\mathrm{cm}$ the superimposed resolution is in the order of millimeters as the computed resolution is zero, which means no matchable measurement positions could be found in a $5\,\mathrm{mm}$ grid. As the relevant proximity for gestures is way above this value, no further efforts were made to reduce grid size. The curve then forms a knee upwards at around $2.5\,\mathrm{cm}$, seems to rise linearly up to $6\,\mathrm{cm}$, and increases much more afterward. While small probe distances can be resolved with a very high resolution, a probe distance of, for instance, $6\,\mathrm{cm}$ cannot be reliably distinguished from a distance of $12\,\mathrm{cm}$.

This corresponds to the previous statement, that capacitive proximity sensing is highly nonlinear and most sensitive in close proximity. Although the range could be increased by increasing the electrode size, this would lead to a bigger form factor which is undesired. Displacements between fingers rarely exceed such ranges anyway. The plots show also that the smaller electrodes pointing downwards have a very similar resolution like the larger electrodes.

6.1.3. Conclusion

The technical evaluation has shown that ambient noise does not influence the measurement. The ring should only be used for interactions where the ring is worn on the finger so that there is a reference mass reducing noise. Direct contact with devices with radio modules or generally high electrical emissions may occasionally result in increased noise in the signal. In practice, however, this should not be a limitation. In the case of on-device interactions, e.g., the gesture execution on the back of a smartphone, it is recommended to check whether the gesture recognition can be disturbed or not.

The measurement of the resolution showed that the effective range reaches up to $2.5\,\mathrm{cm}$. Consequently, at least the influence of the neighboring fingers can be detected. Only the thumb is borderline in detection when the ring is worn on the index finger. This is reasoned because the thumb is directed approximately 90 degrees to the index finger when it is fully spread. However, the field is directed so that it does not perceive lateral influence to reduce interference. How far this influences the recognition of multi-finger gestures and if the effective resolution of $2.5\,\mathrm{cm}$ is enough to enable a wide range of multi-finger interactions will be evaluated in the following sections.

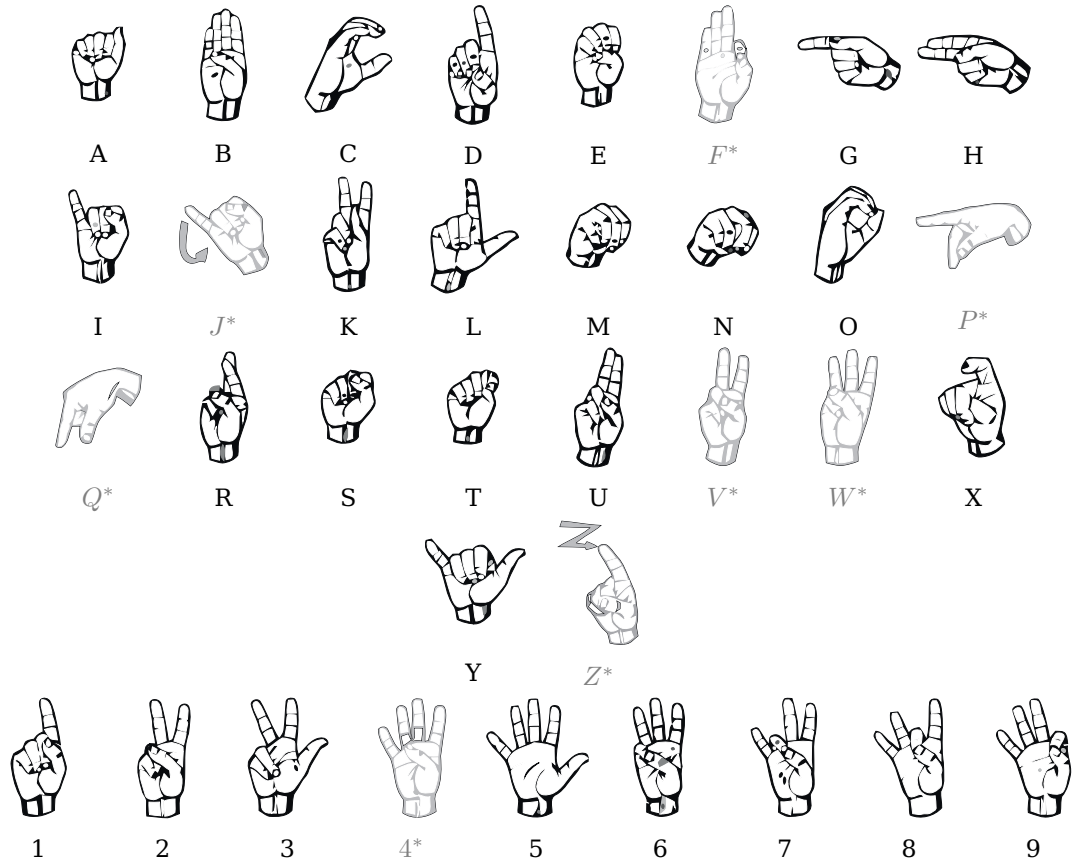


Figure 6.8.: ASL fingerspelling alphabet (A to Z) and numbers (1 to 9). The light gray with an asterisk marked postures were not used for posture interaction evaluation (Images taken from (Marnanel, 2007).).

6.2. Evaluation of Static Interaction Space

In the previous section, the resolution of the capacitive measurement was determined at different distances. This section analyses what these findings imply in practice and which fingers are covered by the measurement. In other words, the static interaction space is determined. It is studied which fingers are in range or not and which implications for the interaction design can be drawn. Also, ring repositioning and user-dependent effects are investigated. As a test set, the American Sign Language (ASL) fingerspelling alphabet (A to Z) and numbers (1 to 9), as shown in Figure 6.8, is utilized. This dataset consists of many different finger poses involving all fingers. This diversity makes the dataset suitable for determining the static interaction space.

6.2.1. Method

This subsection describes the method and setup of this evaluation. It formulates the questions to be answered, the gesture set to be used, the participants, the recording procedure, and finally the analysis procedure.

Questions

In this evaluation, the following questions will be investigated:

Q1 **Static Interaction Space:** What is the effective static interaction space of PeriSense? This involves the following sub-questions: Which fingers are in range of the electric field? Which consequences can be drawn for the interaction design?

Q2 **Repositioning**: Taking on/off the ring can cause slightly differences in ring orientation along the finger axis. Also, during the daily routine, the ring can rotate a little. It should be investigated whether this rotation could influence the measurement or detection in any way.

Q3 **Generalization**: Each hand has different anatomical peculiarities. For example, the fingers are different in length and thickness, and mobility varies. Therefore, it is necessary to investigate whether the measurement data and recognition results can be generalized. Can the data be transferred from one user to another?

Q4 **Confusion Resolution by Additional Sensor Support**: PeriSense also contains an accelerometer as part of the integrated motion sensor. This accelerometer measures the acceleration of the sensor in three orthogonal directions. Since there is no dynamic motion during the ASL postures are performed, only the gravitational force is measured. Therefore, the sensor's orientation to the earth's ground can be determined based on the measured gravitational force vector. Consequently, using the acceleration data as additional information for the ASL postures classification seems obvious. However, do the acceleration data improve the recognition rate?

Gesture Set

As a test set, the American Sign Language (ASL) fingerspelling alphabet (A to Z) and numbers (1 to 9), as shown in Figure 6.8, is utilized. This dataset consists of many different finger poses involving all fingers. This diversity makes the dataset suitable for the determination of the interaction space. The evaluation aims not to detect the ASL alphabet completely with PeriSense, but to determine the interaction space of PeriSense.

Some signs have the same posture but differ in the orientation of the hand (e.g., K and P). These signs do not contribute to interaction space determination because they cannot be detected based only on capacitive sensing. Consequently, these signs are dropped from the test set. Also, signs that include finger movements (e.g., Z) are dropped since they are no static postures. In Figure 6.8, the dropped letters and numbers F, J, P, Q, V, W, Z, and 4 are displayed in light gray and marked with an asterisk. This results in a set of 27 postures to test.

Participants

The experiment was performed with participants that had no previous contact with PeriSense or similar interaction devices and who had the ability to move their fingers painlessly and unrestrainedly. All participants received a voucher for an online store over 30Euro.

Ten colleagues were invited from our institute (female = 4, male = 6, age between 21 and 32) with the following properties of their right hand: hand length between 16.4 and $20.6\,\mathrm{cm}$ ($\mu = 18.36\,\mathrm{cm}$, $\sigma = 1.29\,\mathrm{cm}$), span width of the hand between 18.4 and $23.7\,\mathrm{cm}$ ($\mu = 21.07\,\mathrm{cm}$, $\sigma = 1.46\,\mathrm{cm}$) and index finger length between 6.5 and $8.1\,\mathrm{cm}$ ($\mu = 7.31\,\mathrm{cm}$, $\sigma = 0.45\,\mathrm{cm}$).

The participant's circumference of the index finger was between 56 and $70\,\mathrm{mm}$. To enable evaluation with different finger sizes, three rings were produced with a different inner diameter (18, 20, and $22\,\mathrm{mm}$). The outer sizes kept unchanged and are also identical in regards to technical properties such as signal noise and resolution.

Recording Procedure

Since the index finger is a primary interaction finger, PeriSense was worn on the base segment of the index finger of the right hand. Prior to the tests, the postures were explained and the participants had time to exercise them. The postures were requested and performed in random order. During the experiments, the participants were allowed to take as much time as they needed to take a break, touch and adjust the ring and also relax and move the fingers freely.

The evaluation of the fingerspelling alphabet was split into six sessions distributed over multiple days. In each session, 10 samples of each finger spelling sign were collected. Consequently, each user-provided 1620 samples in total. Every session was initiated by a calibration sequence in which the participants had to perform and hold the respective calibration posture (Figure 6.9) for 5 seconds. This sequence was used to normalize the sensor data.

A maximum of two sessions a day were performed. The participants were able to take a break after the first session. Additionally, the ring battery was changed. This procedure requires taking off the ring. Taking on the ring again probably caused a slightly different ring placement from the first session.

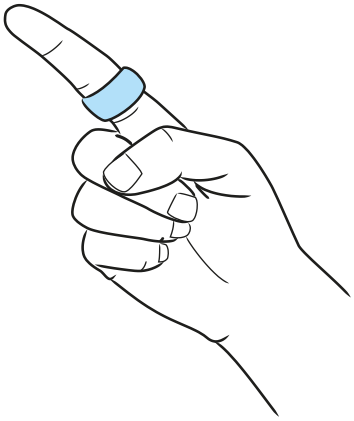


Figure 6.9.: Calibration posture for wearing the ring on the index finger. The posture ensures maximum proximity to the hand or any fingers.

The fingerspelling alphabet to test consists of postures. Postures are indicated by a specific static finger position, although the position is possibly only held for a second. For recording sample data, the participants had to perform the posture shown on the computer screen until it disappeared. The user indicated the start of the posture by pressing a button on the computer keyboard. After this button press, the next $250\,\mathrm{ms}$ of data were labeled with the performed posture by the evaluation software.

All sensor data were logged into a CSV file during the whole session. Besides the four capacitive measurements and the motion data (acceleration, rotation and the ambient magnetic field measurement along the x-, y- and z-axis), the software logged also the associated timestamps (time in milliseconds starting from startup of the ring), sample ids (an increasing posture counter), requested posture labels (when the program requested the participant to perform a certain gesture), posture labels (signs when the user performed the requested posture) and also a temperature measured by the IMU. All measurements were captured at a rate of about $100\,\mathrm{Hz}$.

Analysis Procedure

In order to study the questions Q1 to Q4 from subsection 6.2.1, four tests T1 to T4 were defined as described in the following:

T1 **Test 1 regarding interaction space (Q1)**: Fundamentally, the question is if PeriSense produces distinguishable patterns of the capacitive proximity sensing values even for similar gestures. In order to determine this, a leave one out cross-validation on every single session data set is performed. This reduces possible side effects, e.g., from the repositioning of the ring or from other users (e.g., different hand anatomy, variations in execution, etc.).

T2 **Test 2 regarding repositioning (Q2)**: To study the effect of repeatability and possible effects of repositioning the ring, the sessions of each user are tested against each other. Since six sessions were collected for the fingerspelling alphabet, a 3-fold cross-validation is applied, where two sessions were tested against the remaining four.

T3 **Test 3 regarding generalization (Q3)**: To study the generalization, a leave one user out cross-validation is performed where each user's gestures are tested against the remaining gestures of the other users. This allows us to determine how far different hand anatomy and variations in gesture execution influence the recognition results.

T4 **Test 4 regarding additional sensor support (Q4)**: To study if the acceleration data enhances the recognition results, tests T1 to T3 are repeated under the same conditions but using the acceleration data added to the capacitive sensing values.

The capacitive measurements can vary in different offsets. This can be a result of a constant ring contact with neighboring fingers, in particular for users with shorter fingers, which thereby induces an additional offset. Also, differences in the intrinsic human body capacitance can result in different offsets. Therefore, a calibration was applied in order to normalize the capacitive measurements regarding the offsets. In this process, the mean value of the session's calibration sequence from the whole recorded session. Although all PeriSense's sensor values were logged, the data is constrained to class labels and capacitive measurements for this evaluation. In test T4, the acceleration values were additionally used. For the offset normalized capacitive values and the additional acceleration values in T4, the median value of each labeled posture sequence was determined. This results in a four-element vector for test T1 to T3 and a seven-element vector for T4. For the classification, a Support Vector Machine (SVM) trained with the Sequential Minimal Optimization (SMO) method (Platt, 1998) was applied. In preliminary experiments, different classification algorithms were tested. It was found that SVM offers good performance

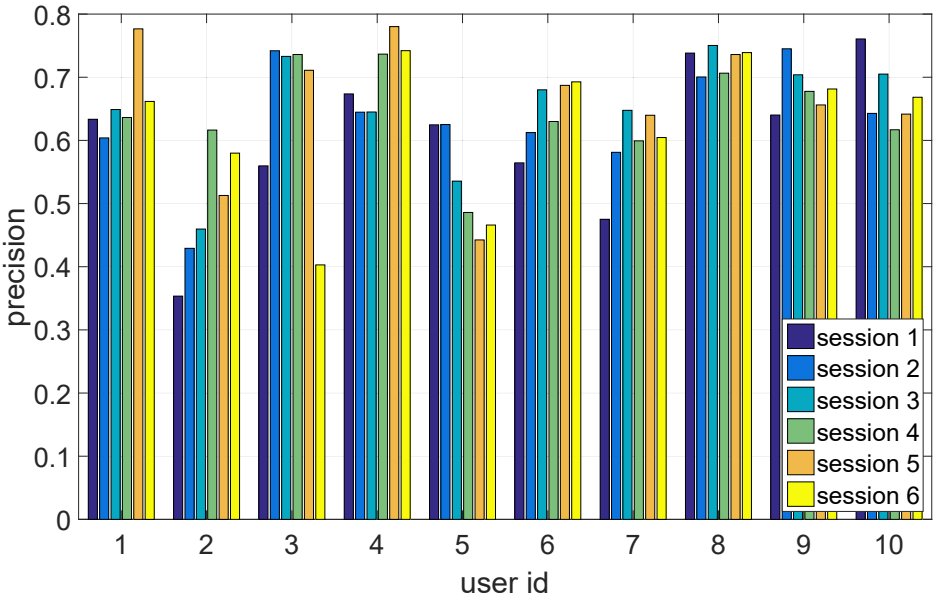


Figure 6.10.: The recognition results of test T1 for each session and user.

over all tests while providing fast training and classification. As a kernel for the SVM, the Pearson VII function-based kernel (PuK) (Üstün et al., 2006) was used. The kernel and regularization parameters were determined for each test by a parameter selection through maximizing the precision. Further, all given rates in the following sections indicate the precision, which is defined as the number of true positives (TP) over the number of true positives and the number of false positives (FP): $Precision = \frac{TP}{TP+FP}$.

6.2.2. Results

In the following, the results of each test are presented.

Results of Test T1

Test T1 performing a leave one out cross-validation on each session results in an average precision over all sessions of 0.64. Figure 6.10 shows each session's precision grouped by the corresponding user ids, and Figure 6.11 shows the histogram of each session's precision. Two sessions even reach 0.78 despite the large and similar set of postures. Eighteen sessions achieve results of 0.7 and more. Twenty-seven sessions reach a precision between 0.6 and 0.7. Fourteen sessions show a precision between 0.4 and 0.6. The lowest achieved precision for a session is 0.35. The

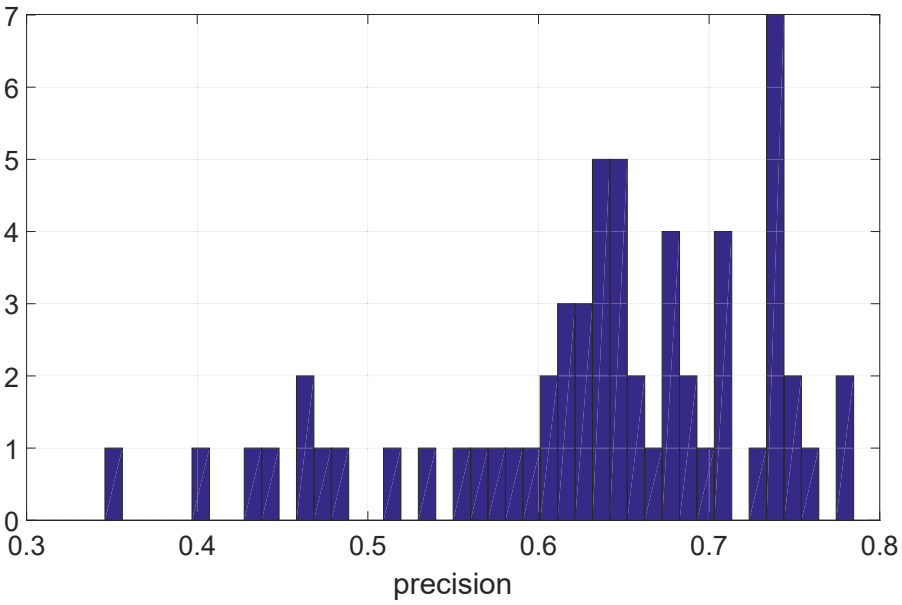


Figure 6.11.: Histogram of the recognition results of test T1.

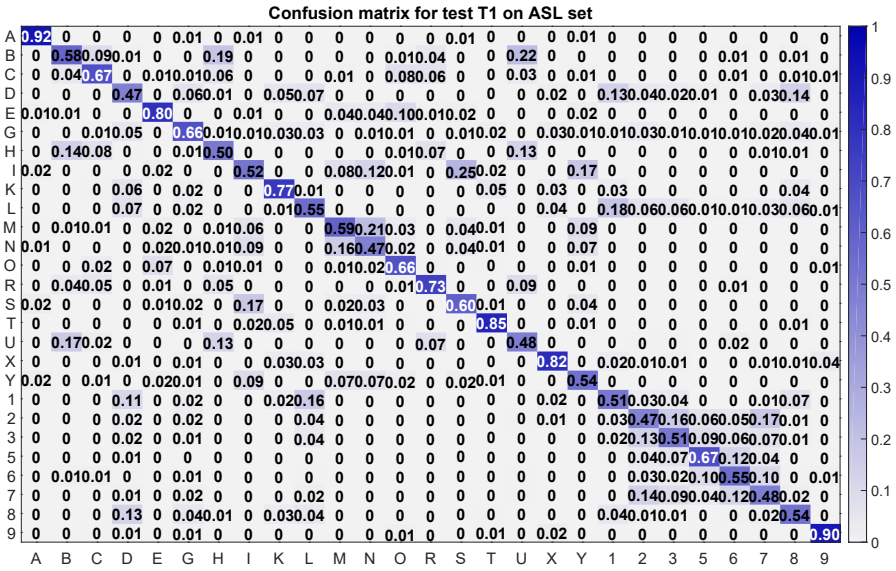


Figure 6.12.: Confusion matrix for test T1 of ASL posture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

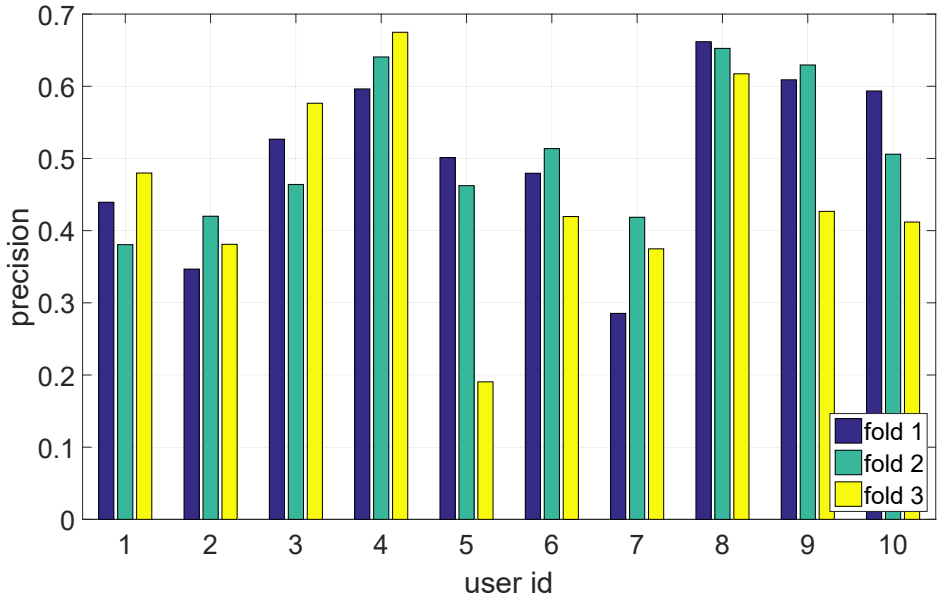


Figure 6.13.: The recognition results of test T2 for each fold and user.

standard deviation over all sessions is 0.1. Seven users reach an average precision over the six sessions of at least 0.6. The standard deviation over the user's average precision is 0.075.

Figure 6.12 depicts the confusion matrix of test T1. It plots how often each performed posture has been confused with other ones. The columns should sum up to one. Since the test set is large and space for plotting is limited, the precision values are rounded to two digits after the point. This causes some columns not to sum up to one. The confusion matrix reveals that the numbers (except 9) are the most difficult to distinguish. Numbers 1, 2, 3, and 8 are also confused with L, D, and sometimes with G and X. The letters M, N, I, S, and Y are also confused. The last confusion group is B, H, and U. The best results are obtained for A, E, T, X, and 9 with a precision of more than 0.79 and the lowest confusion. Overall, no posture is below 0.47.

Results of Test T2

For test T2 performing a 3-fold cross-validation over the six sessions for each user, the precision drops to 0.49. Figure 6.13 shows each fold's precision grouped by the corresponding user ids, and Figure 6.14 shows the histogram of each fold's precision. Eight folds achieves a precision of at least 0.6. The maximum precision reached is 0.67, the lowest is 0.19 and 0.29, and the standard deviation is 0.12.

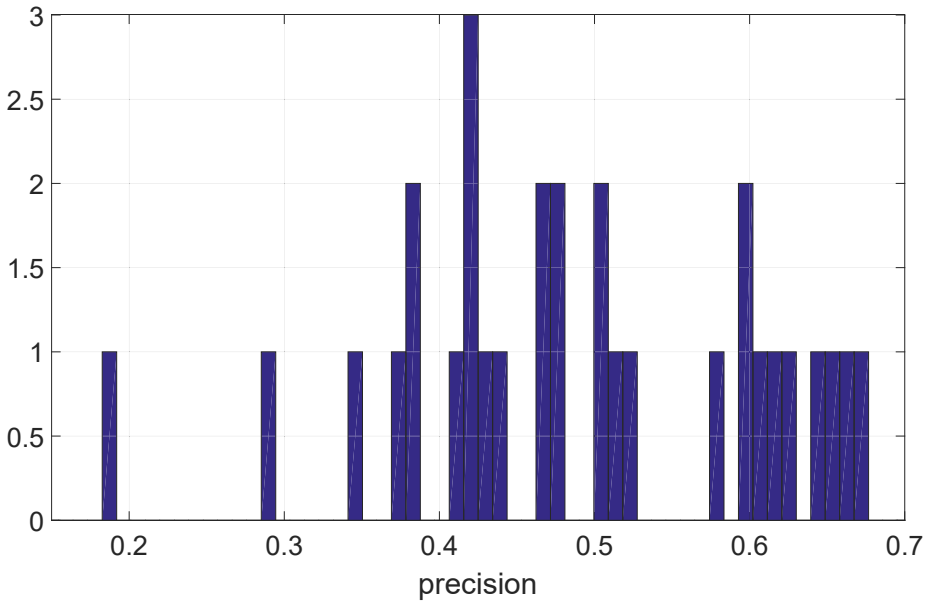


Figure 6.14.: Histogram of the recognition results of test T2.

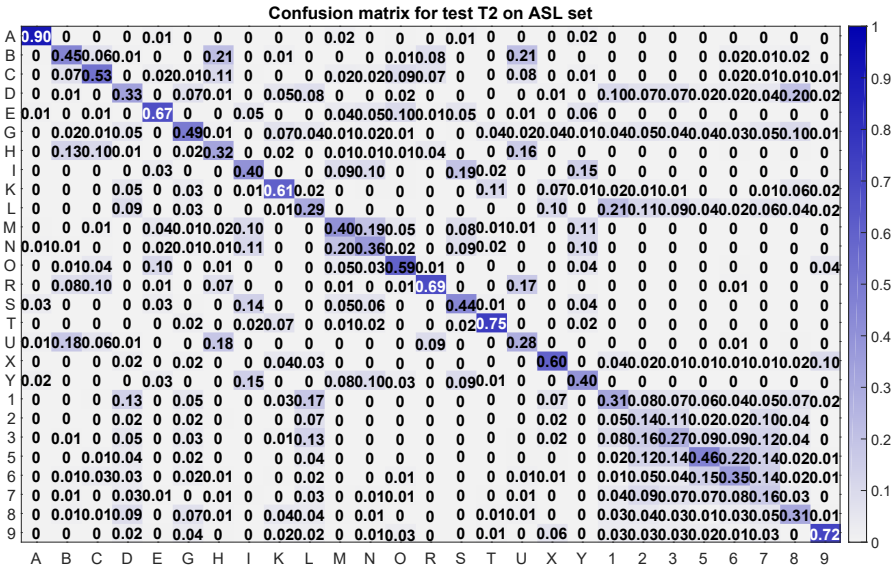


Figure 6.15.: Confusion matrix for test T2 of ASL posture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

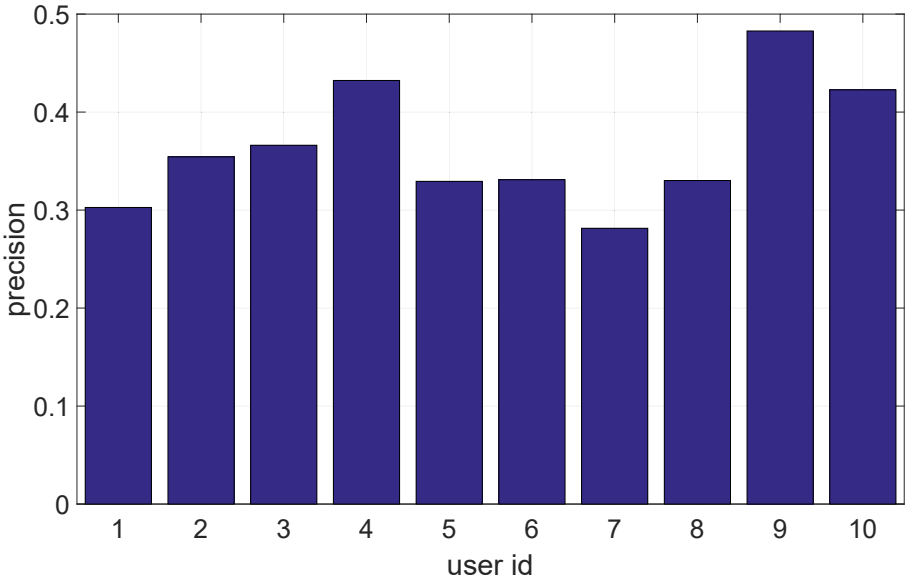


Figure 6.16.: The recognition results of test T3 for each user.

Figure 6.15 depicts the confusion matrix of test T2. It plots how often each performed posture has been confused with other ones. The columns should sum up to one. Since the test set is large and space for plotting is limited, the precision values are rounded to two digits after the point. This causes some columns not to sum up to one. The confusion groups observed in test T1 remain. As in test T1, the letters A, E, T, and 9 also reached the best results with a precision of more than 0.67. In comparison to test T1, X is not anymore in the top five but now letter R. The precision of X drops from 0.82 to 0.6.

Results of Test T3

In test T3 performing a leave one user out cross-validation, the average precision is 0.36. The standard deviation is 0.06. Figure 6.16 shows the precision for each user. Three users archive a precision over 0.4 and the remaining between 0.3 and 0.4 except one resulting with 0.28.

Figure 6.17 depicts the confusion matrix of test T3. It plots how often each performed posture has been confused with other ones. The columns should sum up to one. Since the test set is large and space for plotting is limited, the precision values are rounded to two digits after the point. This causes some columns not to sum up to one. The confusion matrix shows that most postures cannot be distinguished. Only letter A has a precision of 0.89. Letters E, R, T, and 9 have precision between 0.6 and 0.67. The rest performs much worse and shows high confusion.

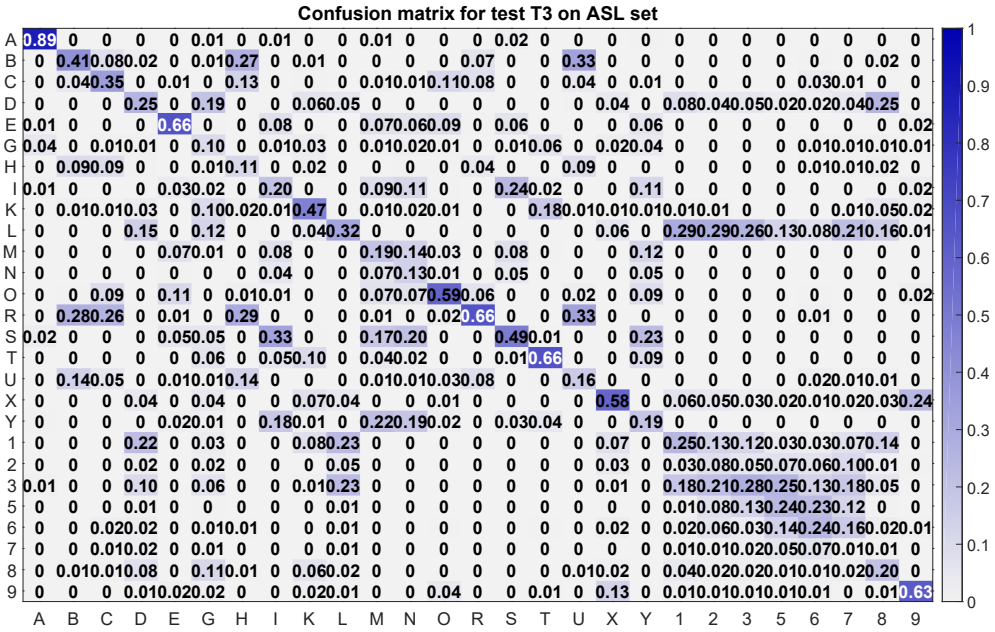


Figure 6.17.: Confusion matrix for test T3 performing a leave one user out cross-validation on the ASL fingerspelling set. The columns refer to the actual classes and the rows to the assigned classes during the classification.

Results of Test T4

In test T4, the tests T1 to T3 were repeated using acceleration values in addition to the capacitive sensing values. Regarding test T1, the average precision over all sessions is 0.7. Figure 6.18 shows the histogram of test T4-1. Two sessions reach more than 0.8. Three sessions achieve a precision of less than 0.6. The standard deviation over all results is 0.053.

Figure 6.19 depicts the confusion matrix of test T4-1. It plots how often each performed posture has been confused with other ones. As in the previous test results, the columns do not necessarily sum to one due to rounding. The confusion matrix reveals that some confusions are resolved or reduced. The numbers (except 9) are still the most difficult to distinguish. However, the numbers are only confused with L and D. The confusion group M, N, I, S, and Y and the group B and U still exist. The best results are obtained for A, G, H, and 9, with a precision of more than 0.9 and the lowest confusion. Postures E, R, and T achieve a rate between 0.8 and 0.9. Overall, no posture is below 0.46.

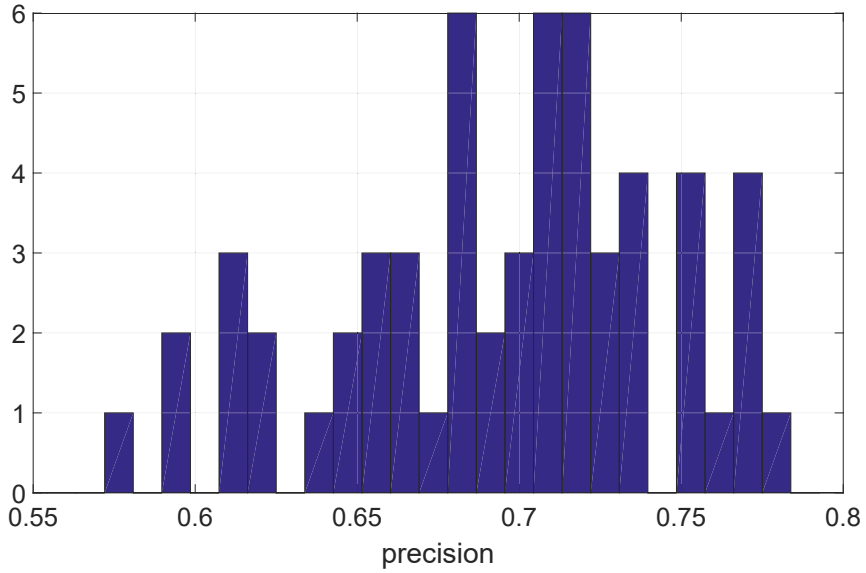


Figure 6.18.: The histogram of the recognition results of test T4-1 performing a leave one out cross-validation on each session.

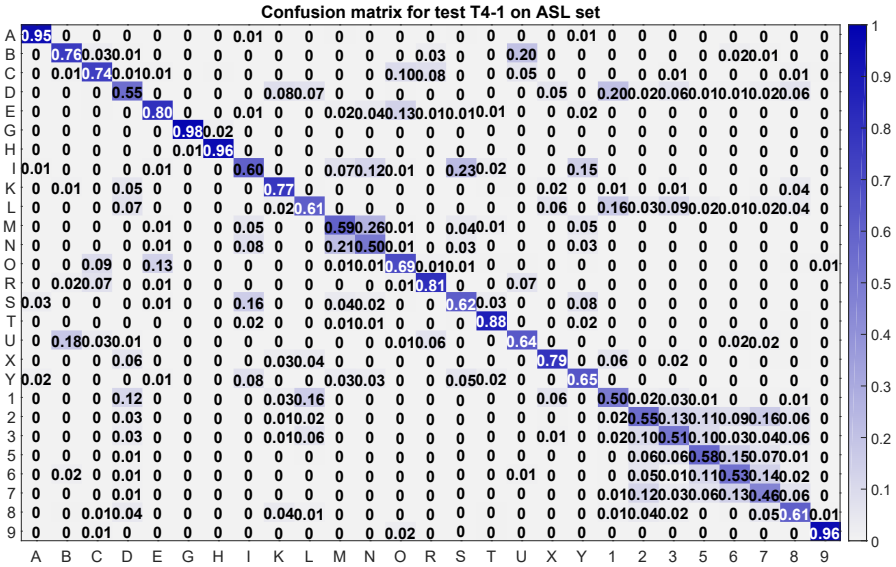


Figure 6.19.: Confusion matrix for test T4-1 of ASL posture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

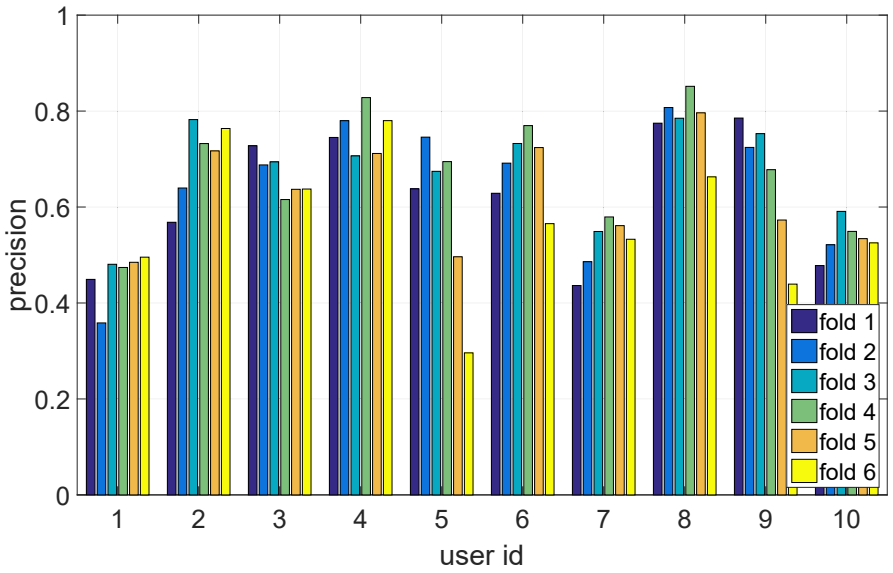


Figure 6.20.: Recognition results of test T4-2 for each fold and user.

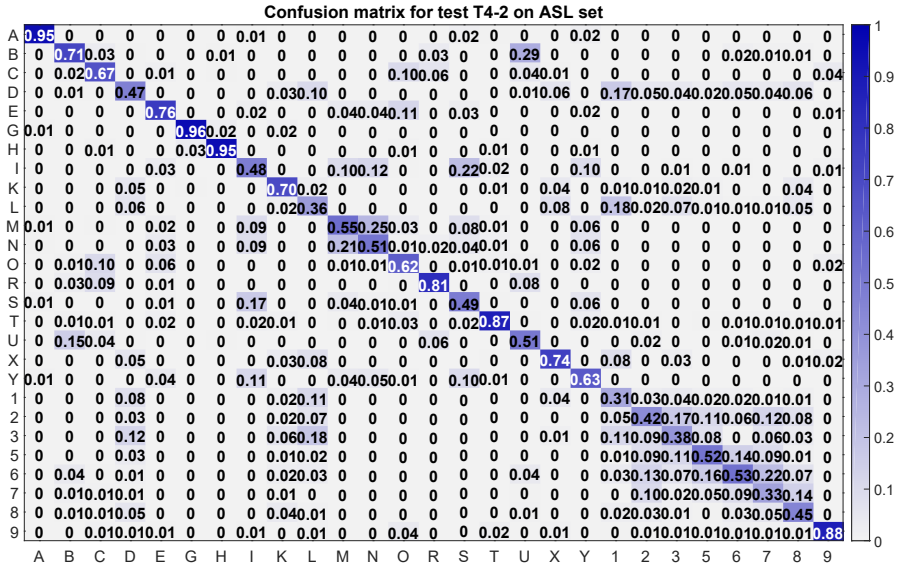


Figure 6.21.: Confusion matrix for test T4-2 of ASL posture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

Regarding test T2, the average precision over all folds is 0.64. Three folds achieve a precision over 0.8 (see Figure 6.20). The lowest fold precision is 0.30 and 0.36, and the standard deviation of the precision for this test is 0.13. The confusion matrix for this test (Figure 6.21) is similar to the confusion matrix of test T4-1. The confusion and precision of postures with higher precision keep almost unchanged. On the other hand, the precision decreases for postures with lower precision and the confusion increases, particularly for the numbers 1 to 8.

Regarding test T3, the average precision over all users is 0.42. Figure 6.22 shows the precision for each user. One user achieves a precision of only 0.14. The highest achieved precision is 0.60, and the standard deviation for this test is 0.12. The confusion matrix (Figure 6.23) shows that most postures cannot be distinguished. Only postures A, G, H, and 9 have a precision of 0.79 and greater. The rest performs much worse and shows high confusion. As already in test T1 and T2, the numbers 1 to 8 perform worst.

6.2.3. Discussion

In test T1, performing a leave one out cross-validation on each session, an average precision of 0.64 was achieved. Eighteen sessions reached a precision over 0.7. In the confusion matrix, two major confusion groups can be identified. The first one is the group M, N, I, S, and Y. These confusions can be explained by the fact that the fingers enclose three sides of the ring, and the palm and these postures vary only in thumb and pinky posture. During the evaluation, it could be observed that participants squeezed their hands with different strengths relating to the postures. This explains for test T1 why some postures, such as I and S, are not entirely confused. Nevertheless, since the electrodes are enclosed by the fingers, it is not possible to detect if the pinkie finger is outstretched or not. It is also difficult to distinguish whether the thumb is outstretched (such in posture Y) or bent and placed on or under the finger (such as in posture I or M). Regarding the signal, a small difference is observable. However, it is too small for reliable distinguishing. Similar issues emerge for the second confusion group: 1, 2, 3, 8, L, D, G, and X. It is difficult to detect reliably if the thumb is outstretched (such as in posture 3 or L) or not (such as in posture D or G). The confusion could be reduced for some postures when the ring would be worn on the thumb. However, this, in turn, increases confusion for other postures. Gestures that differ in the fingers' degree of flexions, such as B, C, E, and S, can be easily distinguished. The same applies to the adjacent fingers. However, the distance between the fingers should not be too large here, as is the case between the thumb and index finger in particular.

In test T2 performing a 3-fold cross-validation over the six sessions for each user, the precision dropped in comparison to test T1. Postures that could be easily distinguished in test T1 achieved a similar precision in test T2. On the other hand, postures with a high confusion in test T1 show a higher confusion in test T2. Con-

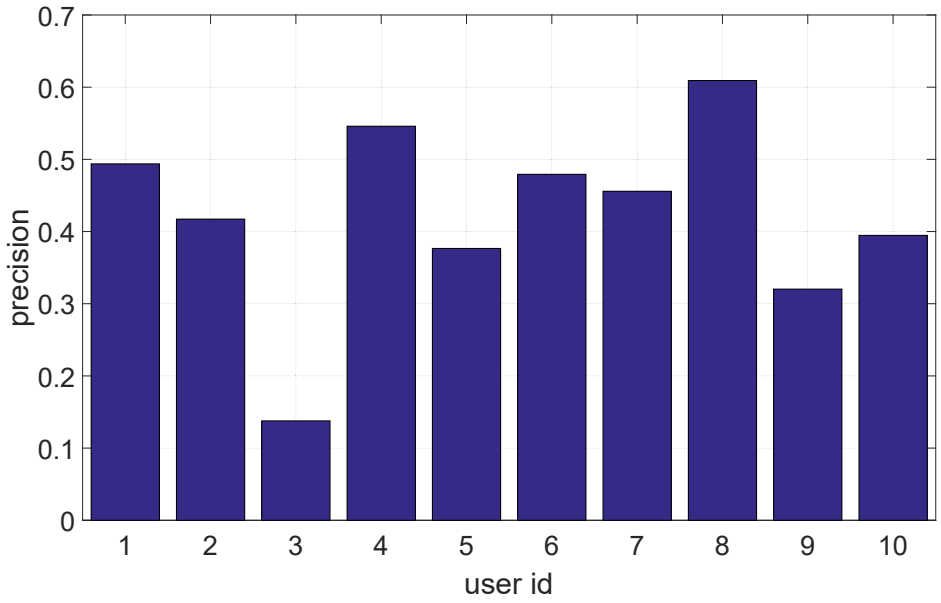


Figure 6.22.: The recognition results of test T4-3 for each user.

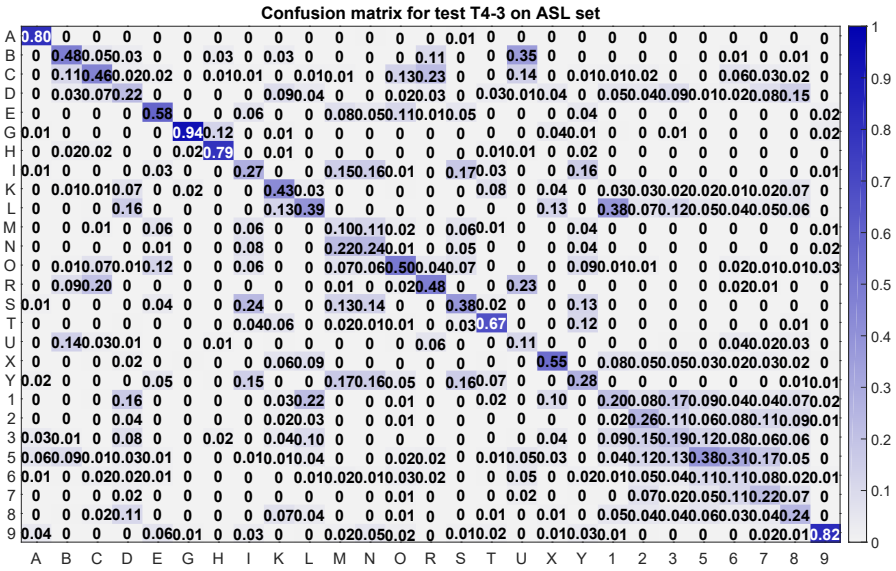


Figure 6.23.: Confusion matrix for test T4-3 of ASL posture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

sequently, possible changes in ring position by reattaching did not influence the recognition result of the proper distinguishable postures.

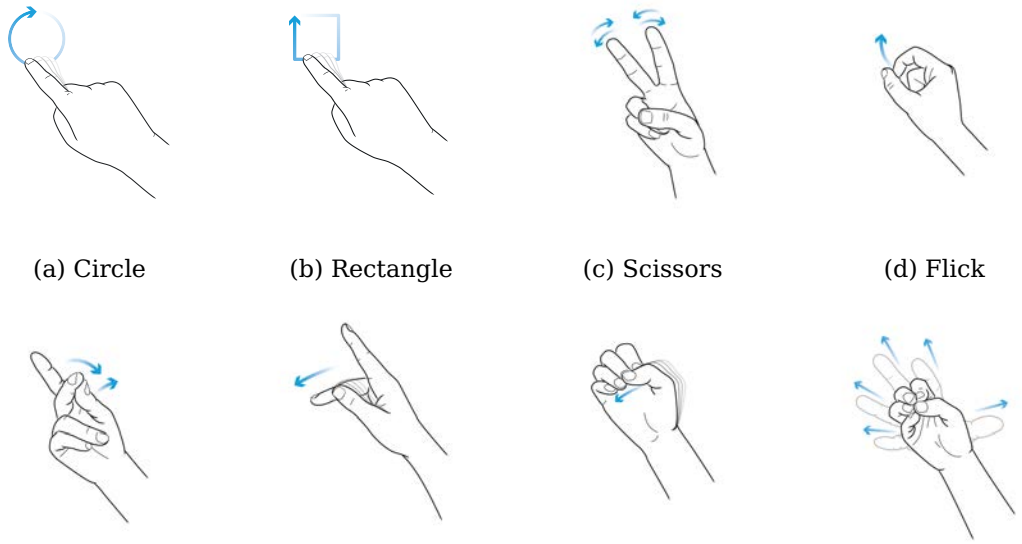
In test T3 performing a leave one user out cross-validation, the average precision dropped by 43.75% in comparison to test T1. The postures are not distinguishable anymore. One issue arises from the execution force of the postures. Some participants squeezed their hands more than others at specific postures such as A and S. This causes slightly different signal levels. Further, it is also to deal with different hand anatomy causing a change of the signal baseline. All these issues together cause a high confusion between some ASL signs. Differences in ring position and orientation, as well as different hand anatomy, should be handled by the calibration. However, it showed that the calibration procedure is not suitable to eliminate these issues adequately.

In test T4 using the accelerometer data additionally for the analysis, some confusions could be resolved and the average overall precision enhanced. Regarding test T1, the precision increased only slightly by 9.83%. However, some confusions could be resolved. Regarding test T2 and T3, the precision increased by 30.61% and 16.67% respectively. In all tests, confusions were reduced.

The two main issues that could be observed are the width of the ring prototype and variations in executing the calibration posture. For users with shorter fingers, the ring already had a little contact with the skin varying with different finger postures. Also, slight variations in the execution of the calibration posture (such as over-stretching the index finger or bending it too much) result in biased calibration sequences. This can cause a change in the ratio between the four signal baselines, and the resulting bias in the pre-processed samples makes it difficult to distinguish similar postures.

Regarding the interaction space, it can be concluded that the thumb is difficult to measure unless it is close to or touching the electrodes. This results from the directed electric field, which is more orthogonal to the electrodes. The thumb, however, usually moves below the ring and thus the electric field. Also, the posture of the pinky finger is difficult to detect.

With an improvement of the calibration process (see also section 6.5), use of the acceleration data, and optimization of the feature selection, a user-specific classifier can be trained on the ASL data set A to Z (without numbers), which achieves good results. However, despite this experiment investigating the interaction space during static finger positions, the dynamic part of the posture, i.e., the change or finger movement between letter transitions, could also be added for classification. This could contribute to an improvement in classification performance.



(a) Circle (b) Rectangle (c) Scissors (d) Flick

(e) Snap (f) Slide Thumb Over Middle Finger (g) Slide Thumb Over Nails (h) Bloom

Figure 6.24.: The multi-finger gesture set used for evaluation of the dynamic interaction space.

6.3. Evaluation of Dynamic Interaction Space

In the previous sections, technical properties were determined, and the static interaction space using postures was studied. This section studies the dynamic interaction space concerning the finger's movement range. Therefore, a finger gesture set characterized by different finger movements is evaluated. Furthermore, it is studied which fingers are in range or not and which implications for the interaction design can be drawn. Also, ring repositioning and user-dependent effects are investigated.

In contrast to the static postures evaluated above, the following gesture set consists of specific finger movements, depicted in Figure 6.24. These finger gestures allow for fine and subtle micro-interactions. Therefore, various gestures different in execution but partially similar to the resulting sensor pattern were selected.

6.3.1. Method

This subsection describes the method and setup of this evaluation. It formulates the questions to be answered, the gesture set to be used, the participants, the recording procedure, and finally the analysis procedure.

Questions

In this evaluation, the following questions will be investigated:

Q1 **Dynamic Interaction Space:** What is the effective dynamic interaction space of PeriSense? This involves the following sub-questions: Which finger movements are in the range of the electric field? Which consequences can be drawn for the interaction design?

Q2 **Repositioning**: Taking on/off the ring can cause slight differences in ring orientation along the finger axis. Also, during the daily routine, the ring can rotate a little. It should be investigated whether this rotation could influence the measurement or detection in any way.

Q3 **Generalization**: Each hand has different anatomical peculiarities. For example, the fingers are different in length and thickness, and mobility varies. Therefore, it is necessary to investigate whether the measurement data and recognition results can be generalized. Can the data be transferred from one user to another?

Q4 **Confusion Resolution by Additional Sensor Support**: PeriSense also contains an accelerometer and a gyroscope as part of the integrated motion sensor. The accelerometer measures the acceleration and the gyroscope the change of rotation of the sensor in three orthogonal directions. Do the acceleration and gyroscope data improve the recognition rate?

Gesture Set

The finger gesture set consists of specific finger movements, depicted in Figure 6.24. These finger gestures allow for fine and subtle micro-interactions. Different finger gestures from various published gesture sets were selected to study the interaction space. These gestures differ in execution but are similar in the resulting sensor pattern. Consequently, it is not intended to show that these gestures can exceptionally be well recognized. Instead, it aims to show where the boundaries are in the differentiation and at the same time which various gestures can be distinguished. Therefore, a mix of gestures has been chosen that, on the one hand, generate similar patterns and, on the other hand, involve different finger movements. This should give an impression of which interactions are possible and which are not. In this manner, gestures *Bloom* and *Flick* have the same index finger movement. Gestures *Circle* and *Rectangle* consist of a very similar pattern. It is also notable that the index finger only draws these two gestures without moving the hand.

Participants

The experiment was performed with participants that had no previous contact with PeriSense or similar interaction devices and who had the ability to move their fingers painlessly and unrestrainedly. All participants received a voucher for an online store over 30Euro.

This group was disjunctive from the evaluation in section 6.2. It involved two invited colleagues and eight external participants acquired via flyers placed at the university campus and supermarkets close-by the campus. Overall, the group consisted of 10 participants with different hand sizes (hand length between 16 and $19.8\,\mathrm{cm}$ ($\mu = 18.26\,\mathrm{cm}$, $\sigma = 1.38\,\mathrm{cm}$), span width of the hand between 18.3 and $23.1\,\mathrm{cm}$ ($\mu = 20.43\,\mathrm{cm}$, $\sigma = 1.56\,\mathrm{cm}$), index finger length between 6.2 and $7.5\,\mathrm{cm}$ ($\mu = 7.07\,\mathrm{cm}$, $\sigma = 0.52\,\mathrm{cm}$)) and gender (female = 5, male = 5, age range between 25 and 46).

The participant's circumference of the index finger was between 56 and $70\,\mathrm{mm}$. To enable evaluation with different finger sizes, three rings were produced with a different inner diameter (18, 20, and $22\,\mathrm{mm}$). The outer sizes kept unchanged and are also identical in regards to technical properties such as signal noise and resolution.

Recording Procedure

Since the index finger is a primary interaction finger, PeriSense was worn on the base segment of the index finger of the right hand. Before these tests, the gestures were explained, and the participants had time to exercise them. The gestures were requested and performed in random order. The participants were asked to move the hand and fingers into the start position of the requested gesture, hold a second, perform the gesture, and keep again a second in the end position of the performed gesture. The short breaks at the beginning and end of each gesture were necessary for the experimenter, who noted the start and end of the gesture by simultaneously pressing a key on a keyboard. The participants received no feedback on the gesture recognition result during the whole experiment. The participants were allowed to take as much time as they needed to take a break, touch and adjust the ring, and relax and move the fingers freely.

The recording of the finger gestures was split into two sessions. In each session, 15 samples of each finger gesture were collected. Consequently, each user provided 240 samples in total. Between the sessions, further multiple data sets of different interaction techniques were recorded. Between each recording, there was a break. No specific interaction technique was recorded in sequence. After every recording session, the ring battery was changed. This procedure requires taking off the ring. Taking on the ring again caused most probably a slightly different ring placement from the previous session.

All sensor data were logged into a CSV file during the whole session. Besides the four capacitive measurements and the motion data (acceleration, rotation and

the ambient magnetic field measurement along the x-, y- and z-axis), the software logged also the associated timestamps (time in milliseconds starting from startup of the ring), sample ids (an increasing posture counter), requested posture labels (when the program requested the participant to perform a certain gesture), posture labels (signs when the user performed the requested posture) and also a temperature measured by the IMU. All measurements were captured at a rate of about $100\,\mathrm{Hz}$.

Analysis Procedure

In order to study the questions Q1 to Q4 from subsection 6.3.1, four tests T1 to T4 were defined as described in the following:

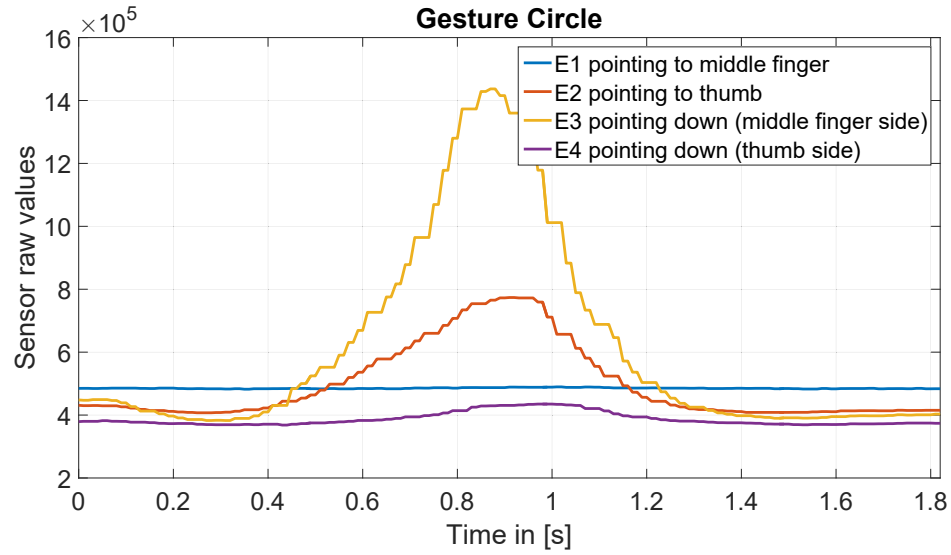
T1 **Test 1 regarding interaction space (Q1)**: The question is if PeriSense produces a distinguishable pattern of the capacitive proximity sensing values even for similar gestures. In order to determine this, a one-leave out cross-validation on every single session data set is performed. This reduces possible side effects, e.g., from the repositioning of the ring or from other users (e.g., different hand anatomy, variations in execution, etc.).

T2 **Test 2 regarding repositioning (Q2)**: The sessions of each user are tested against each other to study the effect of repeatability and possible effects of repositioning the ring. Since two sessions were performed for each user, a 2-fold cross-validation is applied.

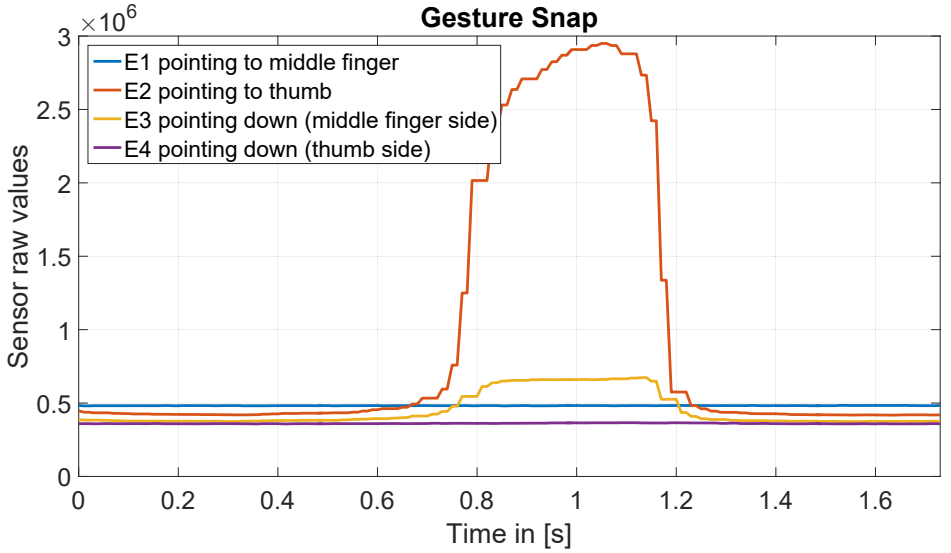
T3 **Test 3 regarding generalization (Q3)**: To study the generalization, a leave one user out cross-validation is performed where each user's gestures are tested against the remaining gestures of the other users. This allows us to determine how far different hand anatomy and variations in gesture execution influence the recognition results.

T4 **Test 4 regarding additional sensor support (Q4)**: To study if the motion sensor data enhances the recognition results, tests T1 to T3 are repeated under the same conditions but using the acceleration and gyroscope data added to the capacitive sensing values.

Although all PeriSense's sensor values were logged, for this evaluation the data is constrained to the class labels and the four capacitive sensor measurements. In test T4, the acceleration and gyroscope values of the motion sensor were additionally used. In the first step, all sessions were min-max normalized in terms of the dimensions. Afterwards all gestures were segmented by the corresponding labels. Since the capacitive measurements can vary in offset, the mean of each gesture was subtracted from the corresponding segmented gesture. Consequently, each normalized and segmented gesture is defined as a multidimensional time series G



(a) Gesture Circle.



(b) Gesture Snap.

Figure 6.25.: Raw values of the capacitive sensor for gestures Circle and Snap.

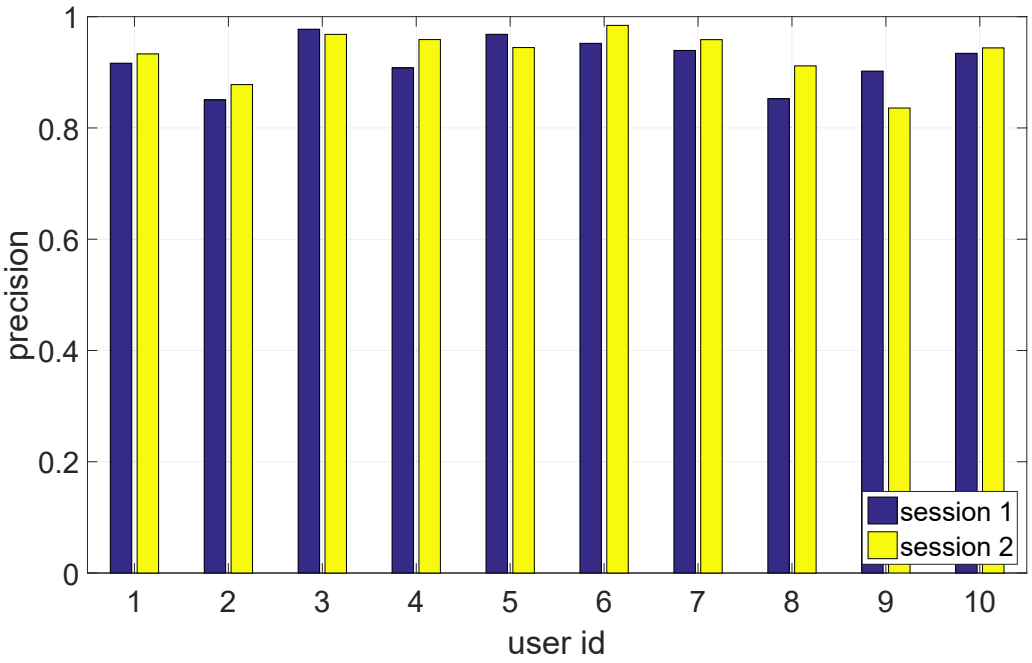


Figure 6.26.: The recognition results of test T1 performing a leave one out cross-validation on each session for each user.

of size n: $G = (g_0, \ldots, g_{n-1})$ with $g_t = (e1_t, e2_t, e3_t, e4_t)$ at time t for test T1 to T3 and with $g_t = (e1_t, e2_t, e3_t, e4_t, ax_t, ay_t, az_t, gx_t, gy_t, gz_t)$ for test T4. The symbols e, a, and g refer to the capacitive measurement, acceleration and the angular velocity respectively.

Figure 6.25 shows exemplary the raw values of the capacitive sensor for gesture *Circle* and *Snap*.

For the classification, a one nearest neighbor (1NN) (ten Holt et al., 2007) classifier was applied. The applied similarity measure is a multidimensional dynamic time warping (DTW) algorithm (ten Holt et al., 2007) and can adapt to different speeds of performing a gesture. The one nearest neighbor classifier using the dynamic time warping distance produces comprehensible and reproducible results, while still being a very robust classifier for sequences (Bagnall et al., 2017).

Further, all given rates in the following sections indicate the precision which is defined as the number of true positives (TP) over the number of true positives and the number of false positives (FP): $Precision = \frac{TP}{TP+FP}$.

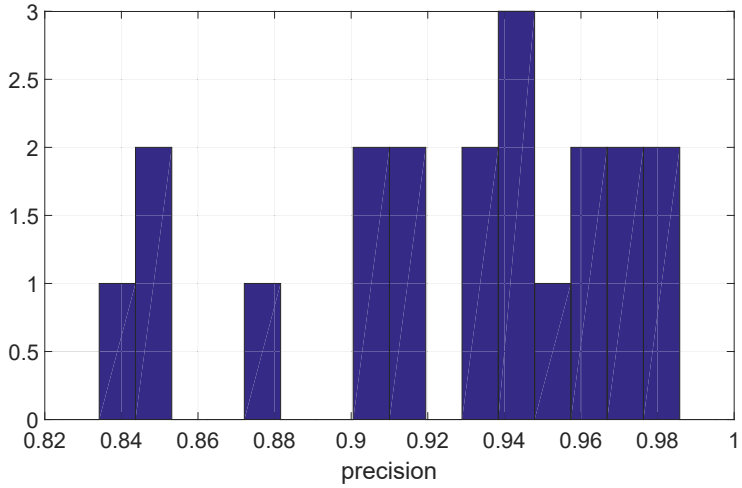


Figure 6.27.: Histogram of the recognition results of test T1.

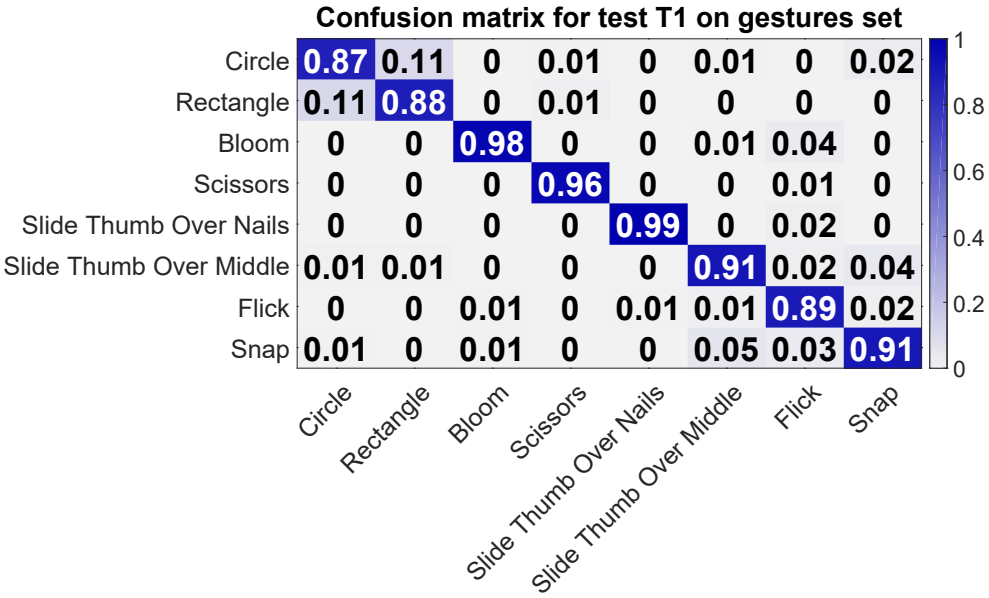


Figure 6.28.: Confusion matrices for test T1 of the gesture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

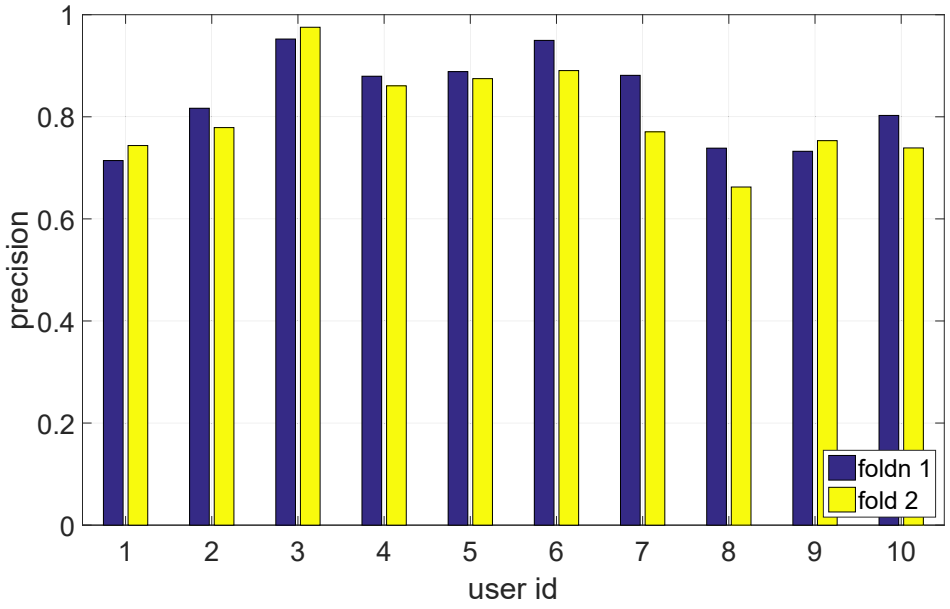


Figure 6.29.: The recognition results of test T2 for each session and user.

6.3.2. Results

In the following, the results of each test are presented.

Results of Test T1

Test T1 performing a leave one out cross-validation on each session results in an average precision over all sessions of 0.93. Figure 6.26 shows each session's precision grouped by the corresponding user ids, and Figure 6.27 shows the histogram of each session's precision. Two sessions reach a precision of 0.98. Only four sessions resulted in a precision less than 0.9 where the minimum precision is 0.84. The standard deviation over all sessions is 0.04. The standard deviation over the user's average precision is also 0.04.

Figure 6.28 depicts the confusion matrix of test T1. It plots how often each performed gesture has been confused with other ones. The columns should sum up to one. Since the test set is large and space for plotting is limited, the precision values are rounded to two digits after the point. This causes some columns not to sum up to one. The confusion matrix reveals that most difficulties occur in the differentiation between Circle and Rectangle. Gestures Flick, Slide Thumb Over Middle, and Snap show some minor confusions with other gestures.

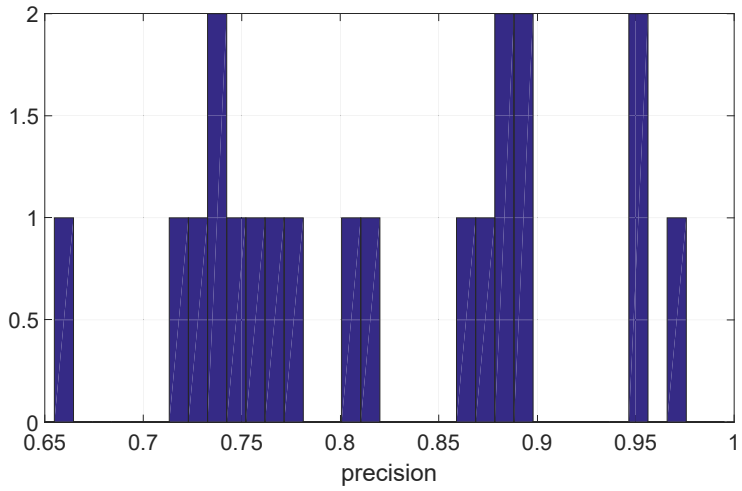


Figure 6.30.: Histogram of the recognition results of test T2.

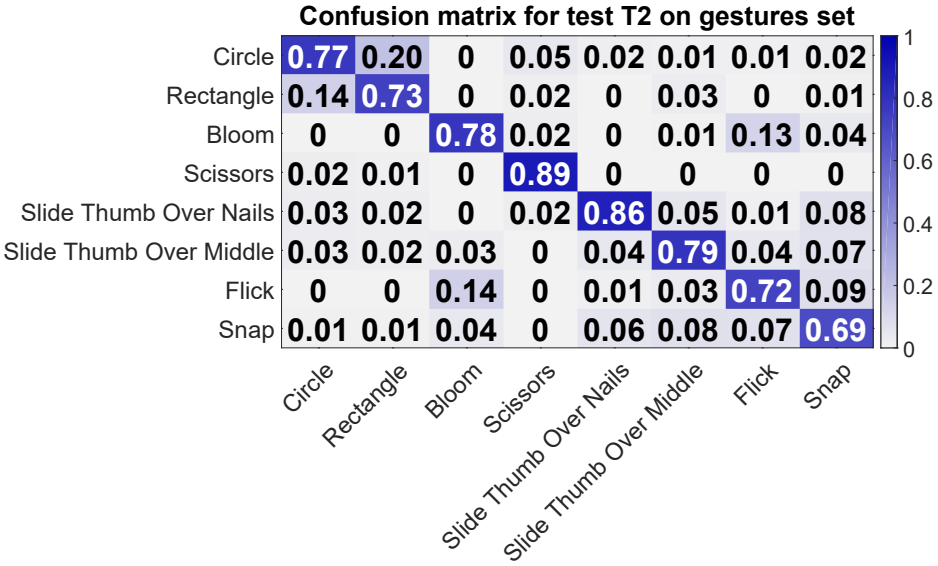


Figure 6.31.: Confusion matrices for test T2 of the gesture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

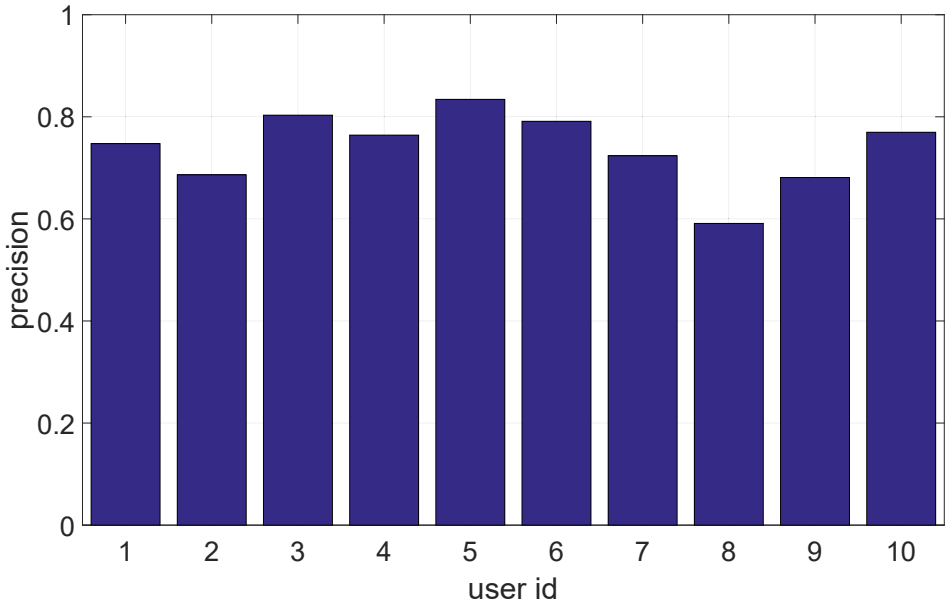


Figure 6.32.: The recognition results of test T3 for each user.

Results of Test T2

For test T2 performing a 2-fold cross-validation over the two sessions for each user, the precision drops down to 0.82. Figure 6.29 shows each fold's precision grouped by the corresponding user ids, and Figure 6.30 shows the histogram of each session's precision. Two folds achieve a precision of 0.95 or higher. Eight folds reach a precision between 0.8 and 0.9. The maximum precision reached is 0.98, the lowest precision is 0.66, and the standard deviation is 0.09.

Figure 6.31 depicts the confusion matrix of test T2. It plots how often each performed gesture has been confused with other ones. The columns should sum up to one. Since the test set is large and space for plotting is limited, the precision values are rounded to two digits after the point. This causes some columns not to sum up to one. The confusions changed in comparison to test T1. As in test T1, there is confusion between Circle and Rectangle. Another confusion is between Bloom and Flick. The precision for gestures Snap, Flick, and Bloom dropped at most.

Results of Test T3

In test T3 performing a leave one user out cross-validation, the average precision is 0.74. The standard deviation is 0.07. Figure 6.32 shows the precision for each user. Two users archive a precision of 0.8 and more. Five user reach a precision between

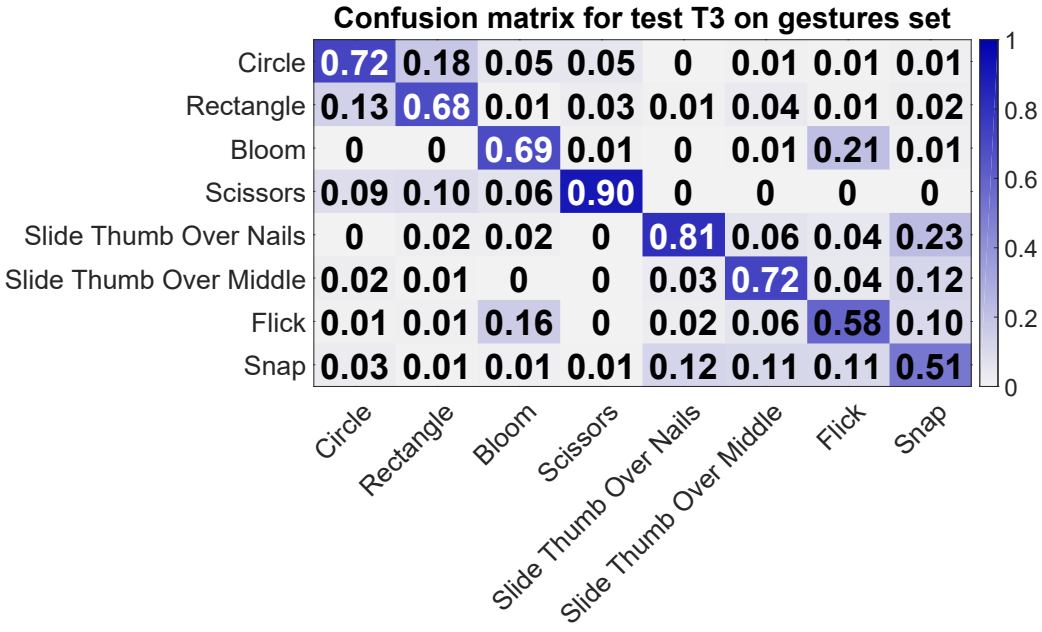


Figure 6.33.: Confusion matrices for test T3 of the gesture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

0.72 and 0.8. Two user achieve a precision of 0.78 and 0.79. The highest precision is 0.83 and the lowest 0.59.

Figure 6.33 depicts the confusion matrix of test T3. It plots how often each performed gesture has been confused with other ones. The columns should sum up to one. Since the test set is large and space for plotting is limited, the precision values are rounded to two digits after the point. This causes some columns not to sum up to one. The confusion matrix is similar to the matrix of test T2. Primarily gestures Flick and Snap dropped more than the remaining in precision. Only gestures Scissors and Slide Thumb Over Nails hardly changed in precision.

Results of Test T4

In test T4, the tests T1 to T3 were repeated using acceleration and gyroscope values in addition to the capacitive sensing values. Regarding test T1, the average precision over all sessions is 0.97. Figure 6.34 shows each session's precision grouped by the corresponding user ids. Four sessions reached a precision of 1.0, three sessions 0.99, and five sessions 0.98. The lowest achieved precision is 0.91. The standard deviation over all results is 0.025. Figure 6.35 depicts the confusion matrix of test T4-1. It plots how often each performed gesture has been confused with other ones. As in

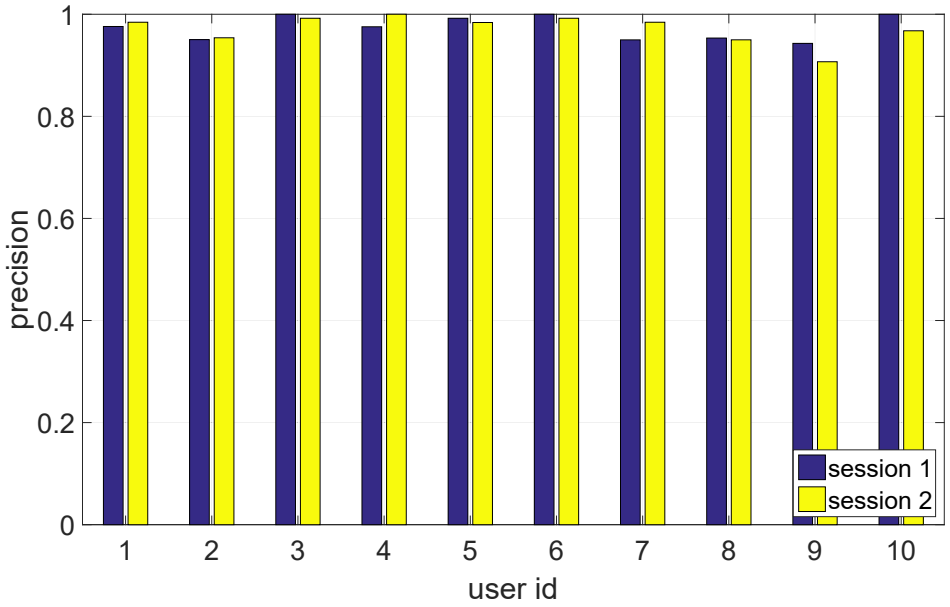


Figure 6.34.: The recognition results of test T4-1 for each session and user.

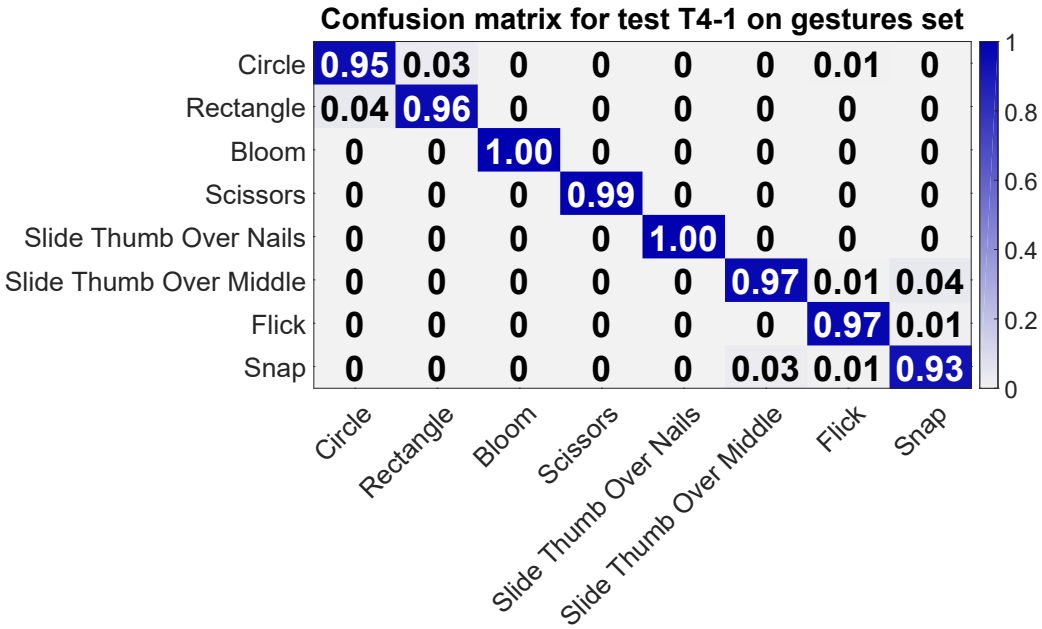


Figure 6.35.: Confusion matrices for test T4-1 of the gesture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

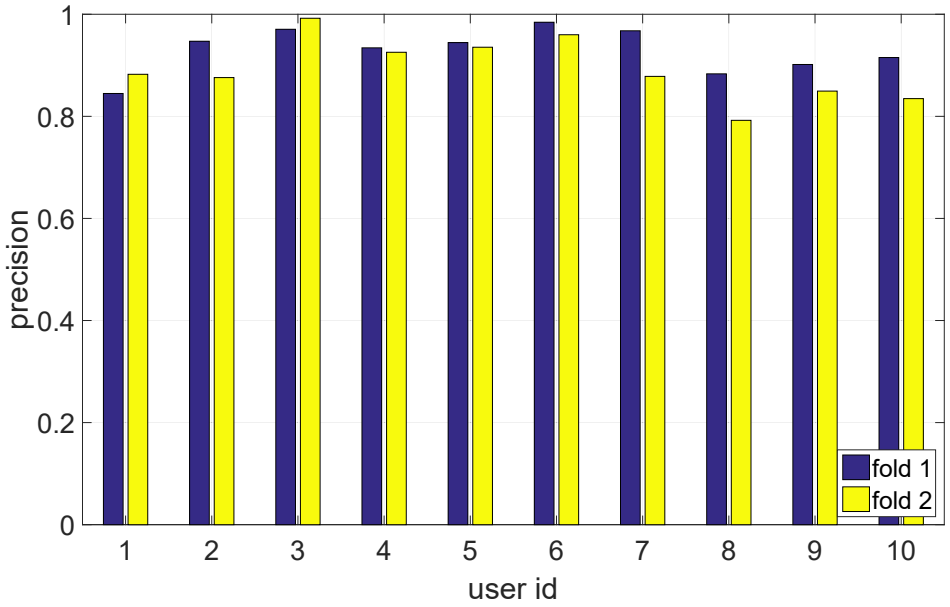


Figure 6.36.: The recognition results of test T4-2 for each session and user.

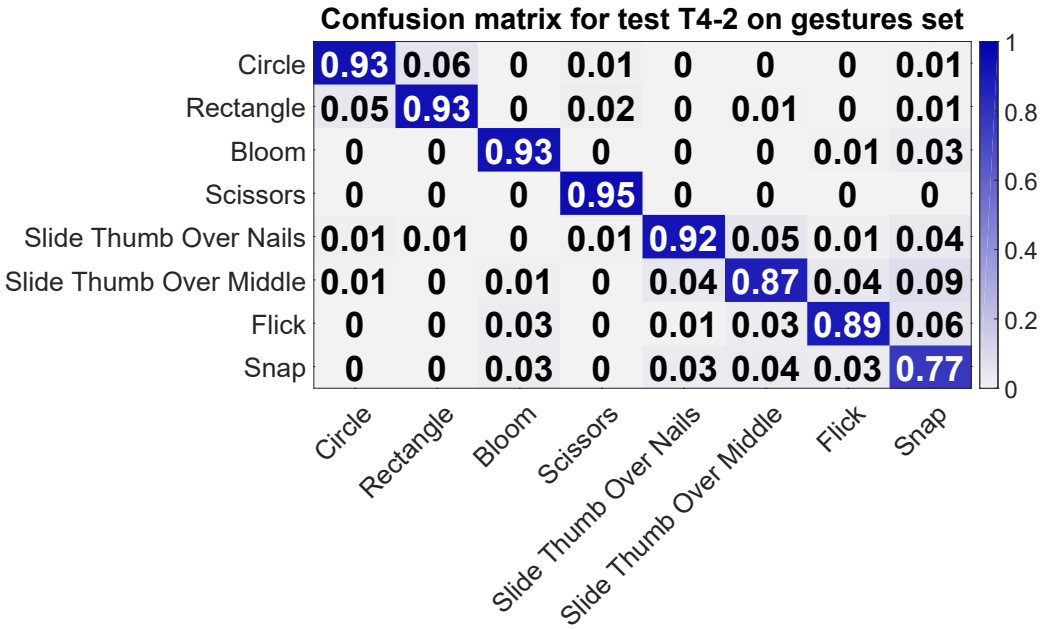


Figure 6.37.: Confusion matrices for test T4-2 of the gesture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

the previous test results, the columns do not necessarily sum to one due to rounding. The confusion matrix reveals that most confusions are resolved or reduced. The confusion group Circle and Rectangle is almost solved. Also, the confusions regarding Slide Thumb Over Middle and Flick are almost resolved. The confusions regarding gesture Snap remain.

Regarding test T2, the average precision over all folds is 0.91. Six folds achieve a precision over 0.94 and eight less than 0.9 (see Figure 6.36). The lowest fold precision is 0.79 and the standard deviation of the precision for this test is 0.053. The confusion matrix for this test (Figure 6.37) shows more confusions than in test T4-1. It shows similar confusion as to the matrix of test T1. Only gesture Snap achieves a low precision of 0.77.

Regarding test T3, the average precision over all users is 0.89. Figure 6.38 shows the precision for each user. The lowest achieved precision is 0.8. Three use achieved a precision of 0.95 and higher, and the standard deviation for this test is 0.048. The confusion matrix (Figure 6.39) changed not significantly in comparison to test T4-2 except for gesture Snap. In comparison to test T3, confusions for all gestures could be significantly reduced.

6.3.3. Discussion

In test T1, performing a leave one out cross-validation on each session, an average precision of 0.93 was achieved. Sixteen of twenty sessions reached a precision higher than 0.9. The confusion matrix reveals only one confusion group between Circle and Rectangle. The similarity of their patterns may explain this confusion. The patterns are primarily distinguished by a short contact of electrodes 2 and 4 with the middle finger. Moreover, most participants reported difficulties in executing these gestures because the execution felt unnatural and required more cognitive attention than the execution of other gestures. This may also have resulted in an unclean execution, which may have increased the likelihood of confusion. Gestures Bloom, Scissors, and Slide Thumb Over Nails provide a robust pattern. Also, gestures Slide Thumb Over Middle Finger and Snap show reasonable results.

In test T2 performing a 2-fold cross-validation over the two sessions for each user, the precision dropped in comparison to test T1. The confusion between Circle and Rectangle increased. Also, the precision for the remaining unistroke gestures decreased visibility. In particular, the performance of the Bloom, Flick, and Snap gestures dropped strongly. This drop may be explained by the similarity of the involved motion of the ring-wearing index finger. The produced signal is fairly similar since the index finger has the same movement in these gestures. Repositioning effects of the ring may increase this effect.

In test T3 performing a leave one user out cross-validation, the average precision dropped by 20.43% in comparison to test T1. The gestures are hardly distinguishable anymore. Only Scissors and Slide Thumb Over Nails show considerable results. The

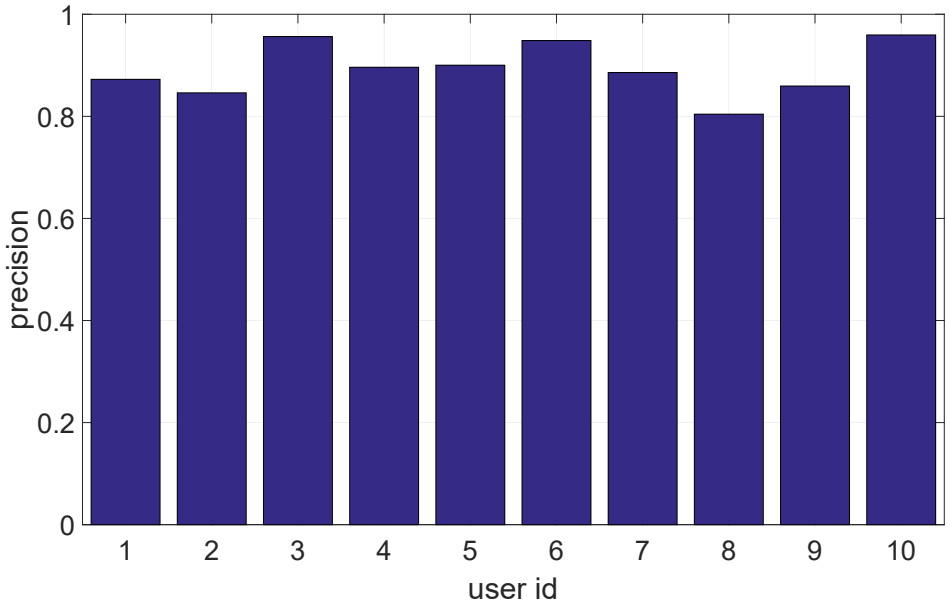


Figure 6.38.: The recognition results of test T4-3 for each user.

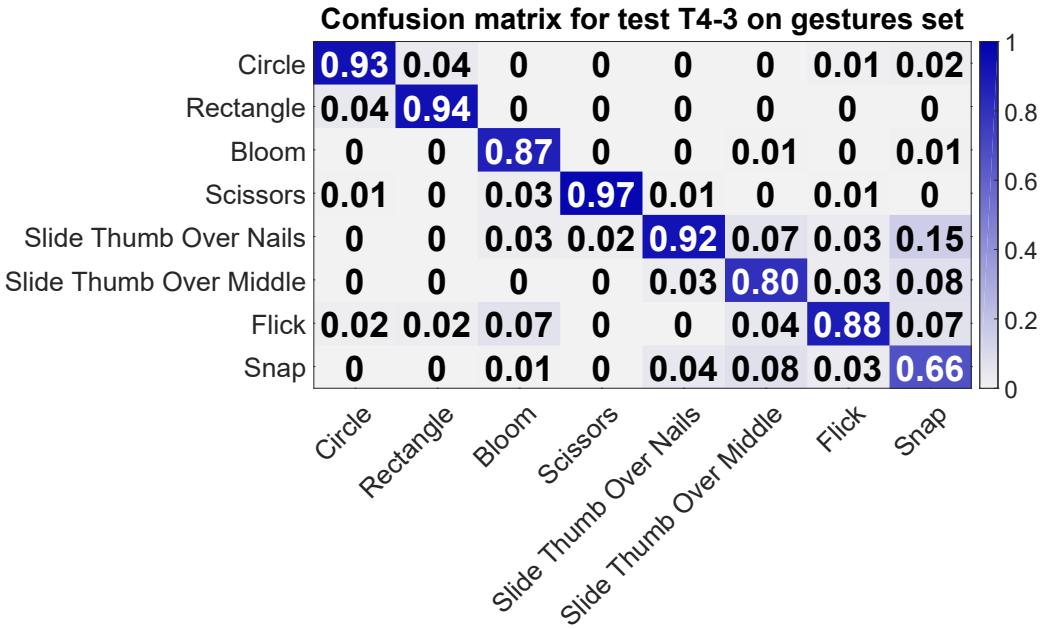


Figure 6.39.: Confusion matrices for test T4-3 of the gesture set. The columns refer to the actual class, the rows to the assigned ones during the classification.

confusion matrix changed hardly in comparison to test T2. Consequently, different finger anatomy and slight differences in ring placement do not seem to influence the precision much as the normalization eliminates different signal baselines and highlights the pattern.

In test T4, using the accelerometer and gyroscope data additionally for the analysis, most confusions could be resolved and the average overall precision enhanced. Regarding test T1, the precision increased only slightly by 4.3%. However, most confusions could be resolved except for Snap. Regarding test T2 and T3, the precision increased by 9.89% and 7.5%, respectively. In all tests, confusions were reduced.

The confusion between Circle and Rectangle was strongly reduced by applying information from the motion sensor because it adds information about directional changes in finger movement. For gesture Rectangle, the finger movement stops at each corner, and for Circle, it is a smooth ongoing movement. Confusion between gestures Flick and Snap comes from variations in the execution of Snap. Sometimes the participants started Snap with index and middle finger straight, bending the middle and moving backward the index finger. This results in almost the same pattern in the capacitive measurements. Again, using the additional acceleration and gyroscope data can solve this confusion issue. Only the confusion between Snap and Slide Thumb Over Middle remains. In general, it can be concluded regarding the interaction space that PeriSense produces proper distinguishable patterns for finger gestures based on capacitive measurements. However, similar motion and variations in execution can lead to confusions. Furthermore, it is to assume that smaller and more electrodes could also reduce confusions. However, the large electrodes seem to hinder the sensing of smaller finger displacements along the electrodes. By employing data from the accelerometer and gyroscope for classification, these confusions can be diminished. This allows a broad definition of finger gestures used with PeriSense enabling various applications.

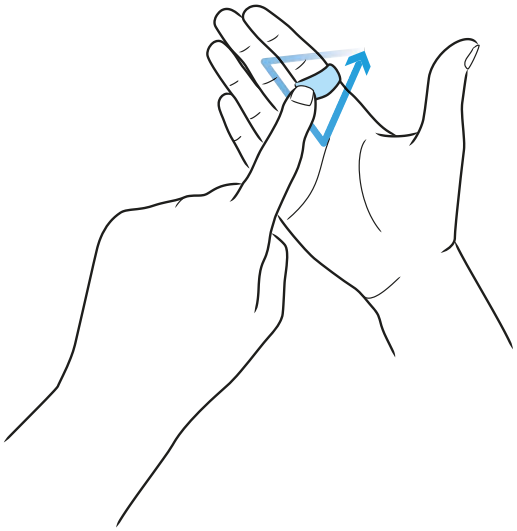


Figure 6.40.: Drawing unistroke gestures above PeriSense.

6.4. Evaluation of Around-Device Interaction Space

In the previous sections, technical properties were studied, and the static and dynamic interaction space were determined. Section 6.1.2 shows that PeriSense can sense the change of conductive objects reliable up to 2.5cm. Consequently, this section evaluates if this capability can be used for around-device interactions extending the interaction space by another interaction technique. For this purpose, it is evaluated if small unistroke gestures drawn with the other hand's index finger above the ring can be detected.

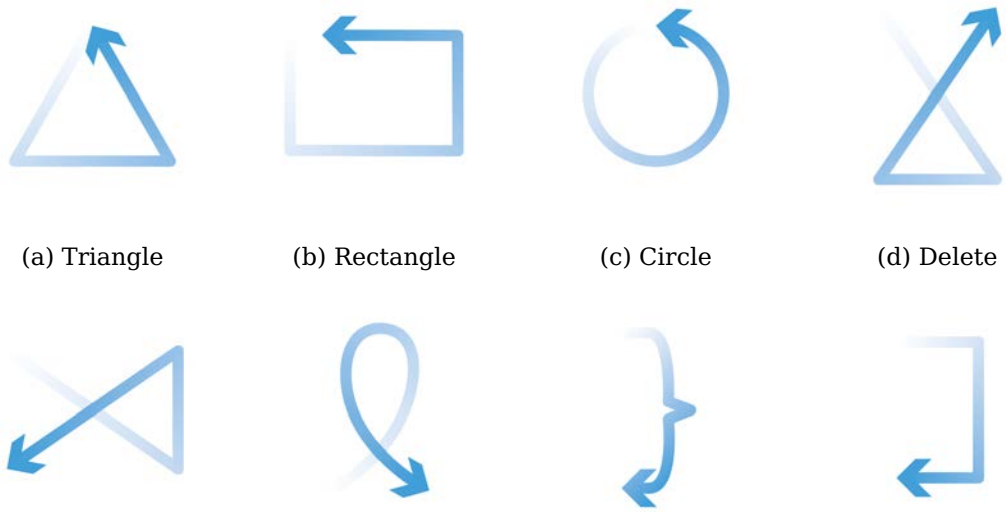
6.4.1. Method

This subsection describes the method and setup of this evaluation. It formulates the questions to be answered, the gesture set to be used, the participants, the recording procedure, and finally the analysis procedure.

Questions

In this evaluation, the following questions will be investigated:

Q1 **Around-Device Interaction Space:** What is the effective interaction space around PeriSense? This involves the following sub-questions: Which finger movements and interactions are detectable? Which consequences can be drawn for the interaction design?



(a) Triangle (b) Rectangle (c) Circle (d) Delete

(e) X (f) Pig Tail (g) Bracket (h) Brace

Figure 6.41.: Unistroke gestures used for evaluation. The gestures start at the thin slightly transparent end, follow the direction of the arrow and end at the arrow.

Q2 **Repositioning**: Taking on/off the ring can cause slight differences in ring orientation along the finger axis. Also, during the daily routine, the ring can rotate a little. It should be investigated whether this rotation could influence the measurement or detection in any way.

Q3 **Generalization**: Each hand has different anatomical peculiarities. For example, the fingers are different in length and thickness, and mobility varies. Therefore, it is necessary to investigate whether the measurement data and recognition results can be generalized. Can the data be transferred from one user to another?

Gesture Set

Since PeriSense measures the capacitance between the electrodes and close surrounding objects, this section evaluates if small unistroke gestures drawn with the other hand's index finger above the ring can be detected. Figure 6.40 depicts an example for such an interaction technique. In order to study how well similar gestures can be distinguished, eight unistroke gestures were selected from (Wobbrock et al., 2007). These are depicted in Figure 6.41.

The unistroke gestures have been chosen, which generate similar patterns. This

set should give an impression which interactions are possible and which are not. In this manner, the unistroke gestures Circle, Rectangle, and Triangle are executed counter-clockwise, Pig Tail is similar to Circle in a different orientation, and Delete is a rotated version of X. Unistrokes Bracket, and Brace are pretty similar to each other too.

These gestures are not directly drawn on the ring surface but at the proximity of about 5 to $30\,\mathrm{mm}$ above the ring. The right hand is to be held as shown in Figure 6.40 and the gestures are drawn with the left hand's index finger where the downward direction of the gesture is towards the palm. The active writing area is directly above the ring. Although most users were right-handed, they reported no difficulties in drawing with their left hand.

Participants

The experiment was performed with participants that had no previous contact with PeriSense or similar interaction devices and who had the ability to move their fingers painlessly and unrestrainedly. All participants received a voucher for an online store over 30Euro.

This group was same as in section 6.3 but disjunctive from the group in section 6.2. It involved two invited colleagues and eight external participants acquired via flyer placed at the university campus and supermarkets close-by the campus. Overall, the group consisted of ten participants with different hand sizes (hand length between 16 and $19.8\,\mathrm{cm}$ (μ = $18.26\,\mathrm{cm}$, σ = $1.38\,\mathrm{cm}$), span width of the hand between 18.3 and $23.1\,\mathrm{cm}$ (μ = $20.43\,\mathrm{cm}$, σ = $1.56\,\mathrm{cm}$), index finger length between 6.2 and $7.5\,\mathrm{cm}$ (μ = $7.07\,\mathrm{cm}$, σ = $0.52\,\mathrm{cm}$)) and gender (female = 5, male = 5, age range between 25 and 46).

The participant's circumference of the index finger was between 56 and $70\,\mathrm{mm}$. To enable evaluation with different finger sizes, three rings were produced with a different inner diameter (18, 20, and $22\,\mathrm{mm}$). The outer sizes kept unchanged and are also identical in regards to technical properties such as signal noise and resolution.

Recording Procedure

Since the index finger is a primary interaction finger, PeriSense was worn on the base segment of the index finger of the right hand. Before these tests, the gestures were explained, and the participants had time to exercise them. The gestures were requested and performed in random order. The participants were asked to move the hand and fingers into the start position of the requested gesture, hold a second, perform the gesture, and keep again a second in the end position of the performed gesture. The short breaks at the beginning and end of each gesture were necessary for the experimenter, who noted the start and end of the gesture by simultaneously pressing a key on a keyboard. The participants received no feedback on the gesture

recognition result during the whole experiment. The participants were allowed to take as much time as they needed to take a break, touch and adjust the ring and also relax and move the fingers freely.

All sensor data were logged into a CSV file during the whole session. Besides the four capacitive measurements and the motion data (acceleration, rotation, and the ambient magnetic field measurement along the x-, y- and z-axis), the software also logged the associated timestamps (time in milliseconds starting from the startup of the ring), sample ids (an increasing posture counter), requested posture labels (when the program requested the participant to perform a particular gesture), posture labels (signs when the user performed the requested posture) and also a temperature measured by the IMU. All measurements were captured at a rate of about 100 Hz.

Analysis Procedure

In order to study the questions Q1 to Q3 from subsection 6.4.1, three tests T1 to T3 were defined as described in the following:

T1 **Test 1 regarding interaction space (Q1)**: The question is if PeriSense produces a distinguishable pattern of the capacitive proximity sensing values even for similar gestures. In order to determine this, a one-leave out cross-validation on every single session data set is performed. This reduces possible side effects, e.g., from the repositioning of the ring or from other users (e.g., different hand anatomy, variations in execution, etc.).

T2 **Test 2 regarding repositioning (Q2)**: To study the effect of repeatability and possible effects of repositioning the ring, the sessions of each user are tested against each other. Since two recording sessions with each user were performed for the unistroke set, a 2-fold cross-validation is applied, where two sessions were tested against each other.

T3 **Test 3 regarding generalization (Q3)**: To study the generalization of the result, a leave one user out cross-validation is performed where each user's gestures are tested against the remaining gestures of the other users. This allows us to determine how far different hand anatomy and variations in gesture execution influence the recognition results.

Since the right hand wearing PeriSense is not moving during the unistroke execution, the capacitive measurements without motion data was only used for the analysis. Additionally, the data of the right electrode was omitted because the middle finger covers it. Consequently, the electrode measurements of the remaining three electrodes were only used for the unistroke gesture recognition. Due to variances in proximity between PeriSense and the drawing finger, each sample was

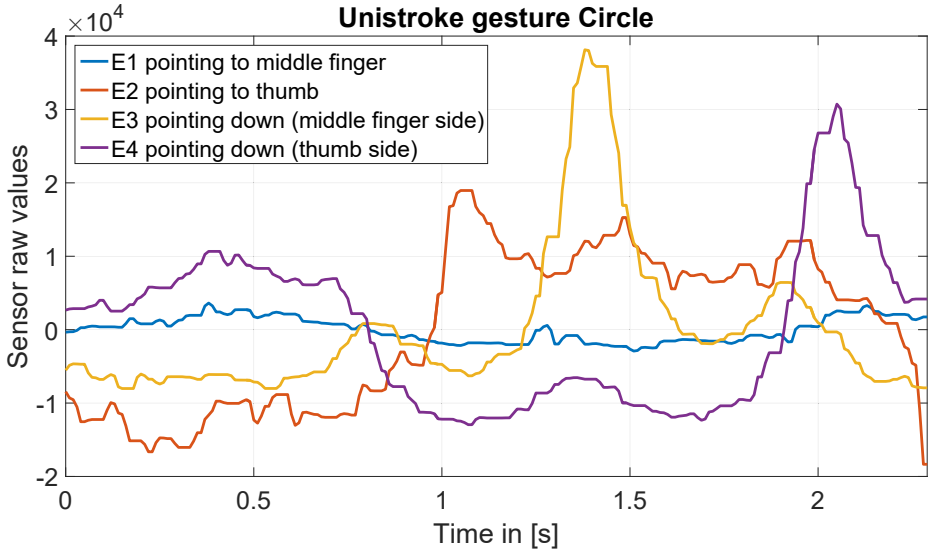


Figure 6.42.: Raw values of the capacitive sensor for unistroke gesture Circle.

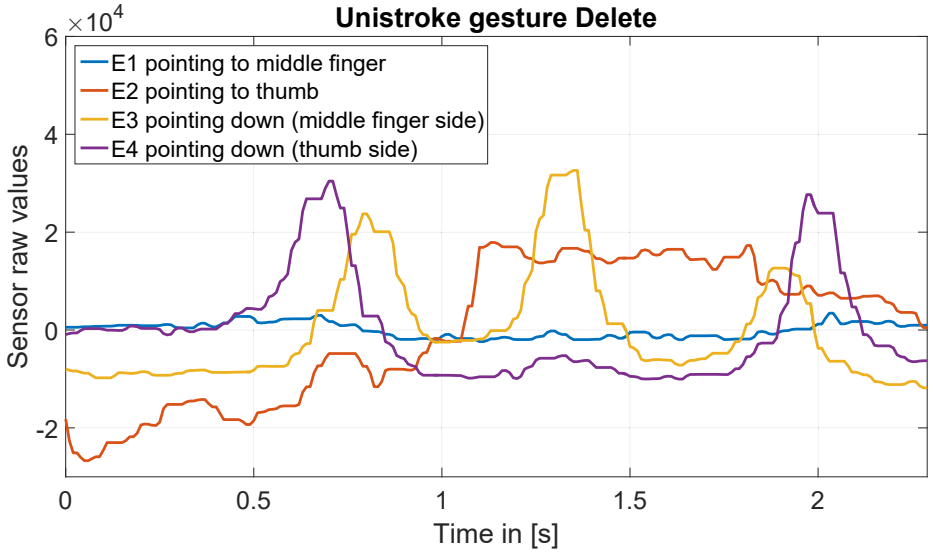


Figure 6.43.: Raw values of the capacitive sensor for unistroke gesture Delete.

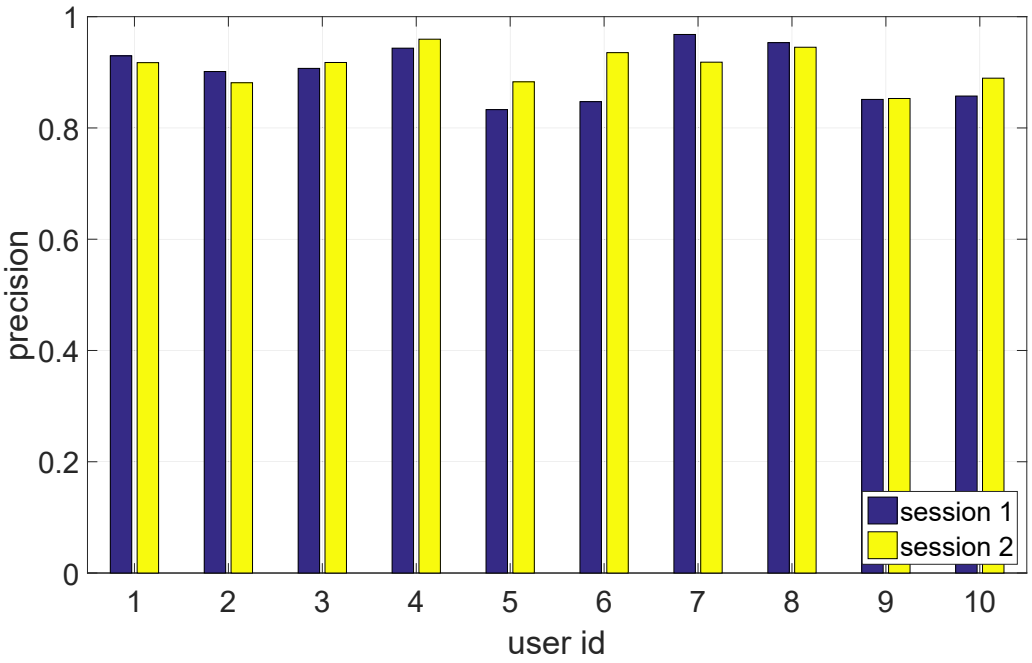


Figure 6.44.: The recognition results of test T1 for each session and user.

standardized by centering to the mean and scaling to unit variance to stress the pattern and normalize its amplitude. Consequently, each standardized unistroke gesture is defined as a multidimensional time series G of size n: $G = (g_0, \ldots, g_{n-1})$ with $g_t = (e1_t, e2_t, e3_t)$ at time t. The symbol e refers to the capacitive measurement.

Figures 6.42 and 6.43 show exemplary the raw values of the capacitive sensor for unistroke gesture Circle and Delete. Since the offsets of the sensor values vary much more than the amplitudes, the mean value for each sensor value was subtracted to make the amplitude visible and comparable.

For the classification, a one nearest neighbor (1NN) (ten Holt et al., 2007) classifier was applied. The applied similarity measure is a multidimensional dynamic time warping (DTW) algorithm (ten Holt et al., 2007) and can adapt to different speeds of performing a gesture. 1NN with DTW distance produces comprehensible and reproducible results while still being a very robust classifier for sequences (Bagnall et al., 2017). Further, all given rates in the following sections indicate the precision, which is defined as the number of true positives (TP) over the number of true positives and the number of false positives (FP): $Precision = \frac{TP}{TP+FP}$.

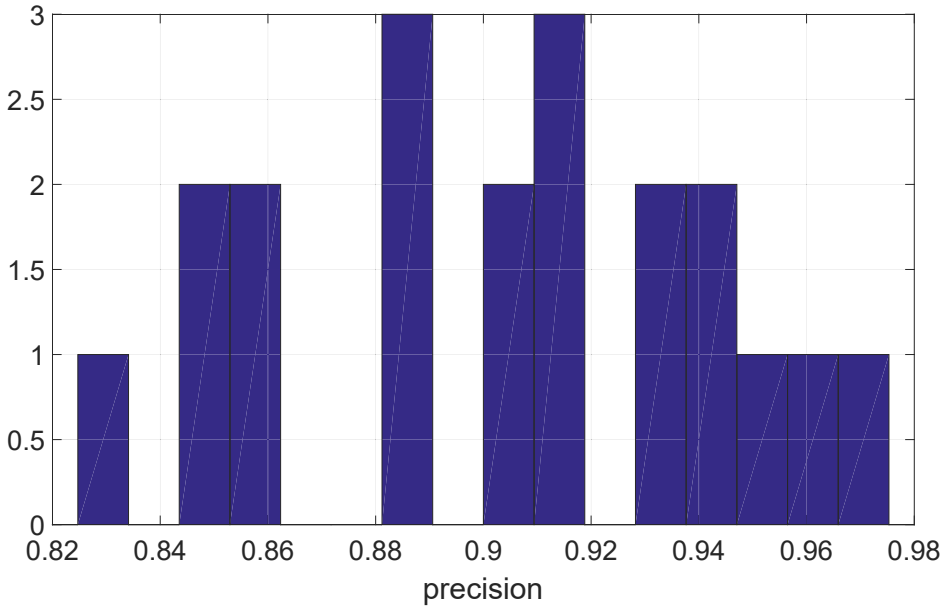


Figure 6.45.: Histogram of the recognition results of test T1.

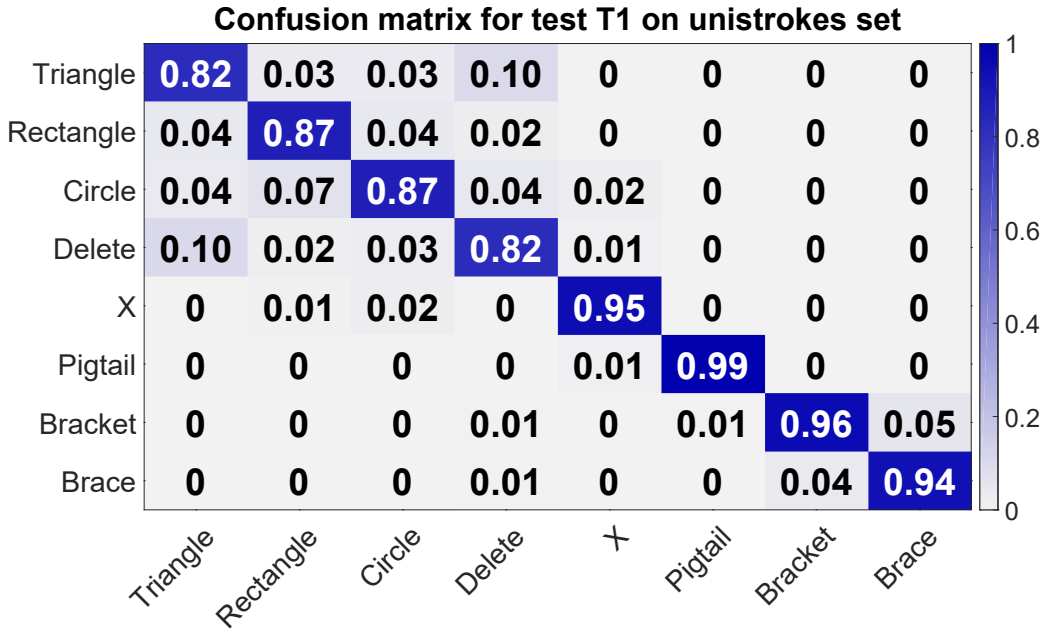


Figure 6.46.: Confusion matrices for test T1 of the unistroke set. The columns refer to the actual class, the rows to the assigned ones during the classification.

6.4.2. Results

In the following, the results of each test are presented.

Results of Test T1

Test T1 performing a leave one out cross-validation on each session results in an average precision over all sessions of 0.9. Figure 6.44 shows each session's precision grouped by the corresponding user ids, and Figure 6.45 shows the histogram of each session's precision. Four sessions reach a precision of 0.95 and more. Twelve sessions resulted in a precision of 0.9 and higher. The minimum precision is 0.83. The standard deviation over all sessions is 0.04. The standard deviation over the user's average precision is also 0.04.

Figure 6.46 depicts the confusion matrix of test T1. It plots how often each performed gesture has been confused with other ones. The columns should sum up to one. Since the test set is large and space for plotting is limited, the precision values are rounded to two digits after the point. This causes some columns not to sum up to one. The confusion matrix shows a reasonable accuracy of0.94 and more for gestures Bracket, Brace, Pig Tail, and X. Despite the good precision of Bracket and Brace, there is some confusion between these two unistrokes. A big confusion cluster can be identified for Triangle, Rectangle, Circle, and Delete. All of these gestures have a precision below 0.88. Also, X is sometimes confused with the gestures of this group.

Results of Test T2

In test T2 performing a 2-fold cross-validation over the two sessions for each user, the precision drops down to 0.86. Figure 6.47 shows each fold's precision grouped by the corresponding user ids, and Figure 6.48 shows the histogram of each session's precision. Two folds achieve a precision of 0.97. Eight folds reach a precision of $0.9$and higher. The maximum precision reached is 0.97, the lowest precision is 0.41 and 0.63, and the standard deviation is 0.13. One user shows a low result on both folds.

Figure 6.49 depicts the confusion matrix of test T2. It plots how often each performed unistroke has been confused with other ones. As in the previous test results, the columns do not necessarily sum to one due to rounding. Overall, the confusion groups remain unchanged in comparison to test T1. As in test T1, there is confusion between Bracket, Brace, Pig Tail, and X. There are some more confusions regarding Brace, Bracket, and Circle in general.

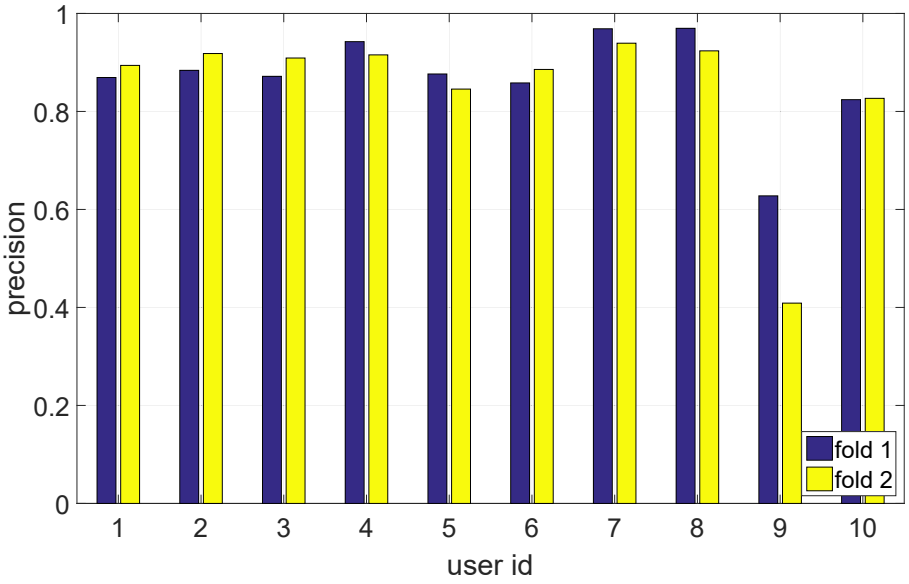


Figure 6.47.: The recognition results of test T2 for each session and user.

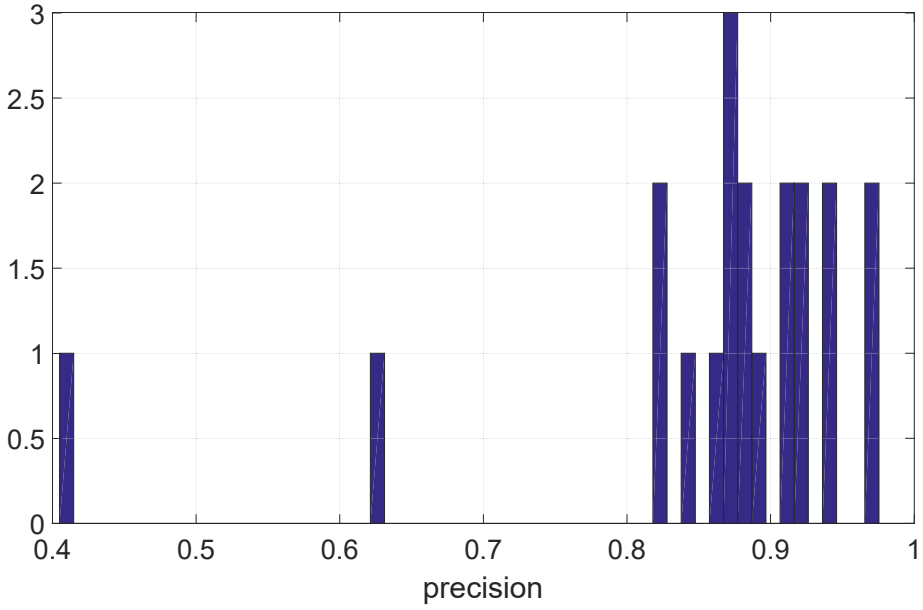


Figure 6.48.: Histogram of the recognition results of test T2.

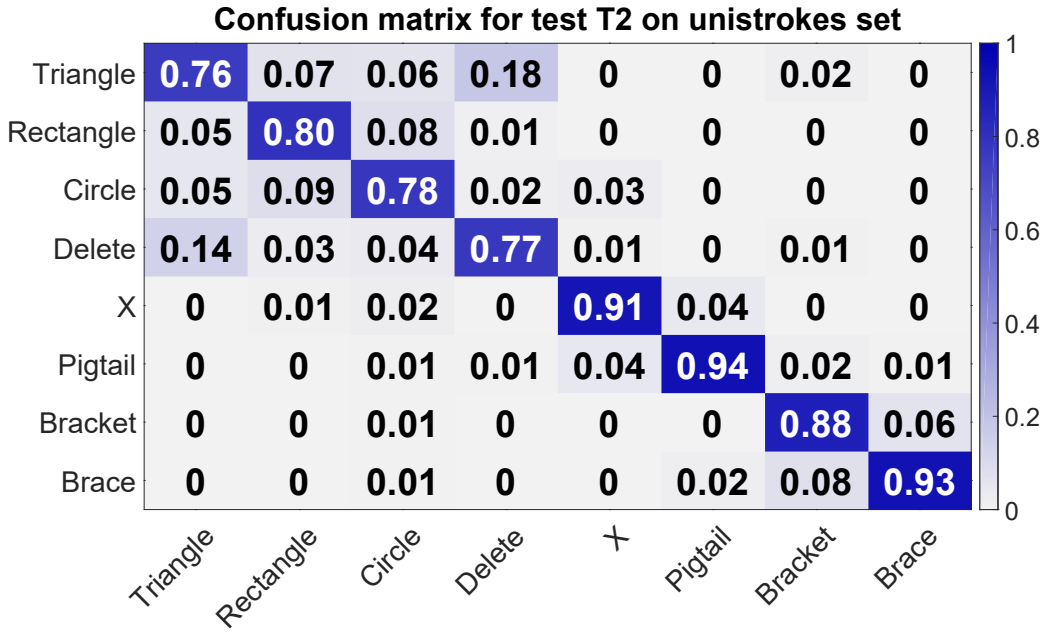


Figure 6.49.: Confusion matrices for test T2 of the unistroke set. The columns refer to the actual class, the rows to the assigned ones during the classification.

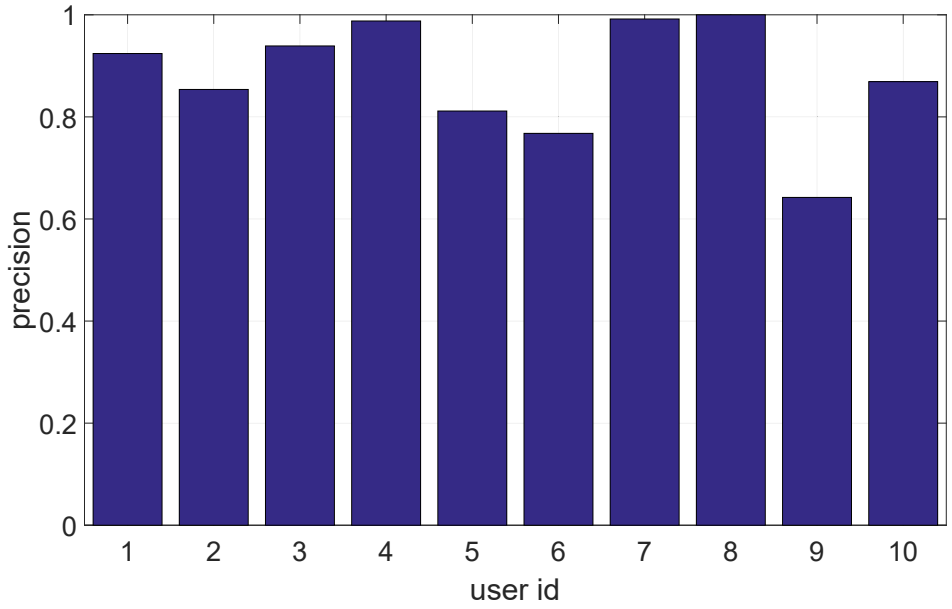


Figure 6.50.: The recognition results of test T3 for each user.

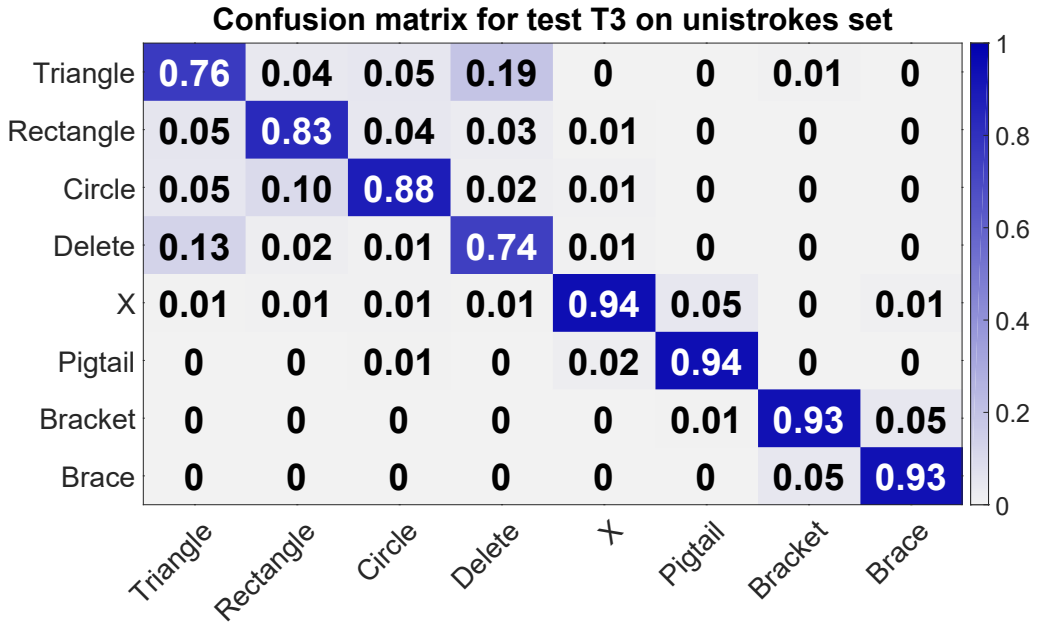


Figure 6.51.: Confusion matrices for test T3 of the unistroke set. The columns refer to the actual class, the rows to the assigned ones during the classification.

Results of Test T3

In test T3 performing a leave one user out cross-validation, the average precision is 0.88. The standard deviation is 0.11. Figure 6.50 shows the precision for each user. For one user, a precision of 1.0 and for two users a precision of 0.99 is achieved. Five users reach a precision of 0.92 or higher. Four users achieve a precision between 0.77 and 0.88. The same user as in test T2 also achieves in test T3 the lowest precision of 0.64.

Figure 6.51 depicts the confusion matrix of test T3. It plots how often each performed gesture has been confused with other ones. As in the previous test results, the columns do not necessarily sum to one due to rounding. The confusion matrix is more similar to test T1 than to test T2.

6.4.3. Discussion

The test revealed that PeriSense could also detect around-device interactions performed with the other hand, in this concrete case, two-handed unistroke gesture input. In general, this input technique generates well distinguishable patterns for most gestures. The big confusion group of Triangle, Rectangle, Circle, and Delete seems to arise from the one-dimensional electrode arrangement, particularly for the

confusion between Triangle and Delete since it is not possible to detect the direction of movement along the electrode.

There is no significant difference between user dependence and user independence tests. Changes in ring orientation are no issue since the ring is fixed between the index and middle finger. Only for one user out of 10 a low result was obtained. Judging from the signals, in this case, it seems that the gestures were not performed exactly over the ring.

The ability to detect multiple finger gestures and around-device interactions increases the interaction space and, thus, the variations of applications.

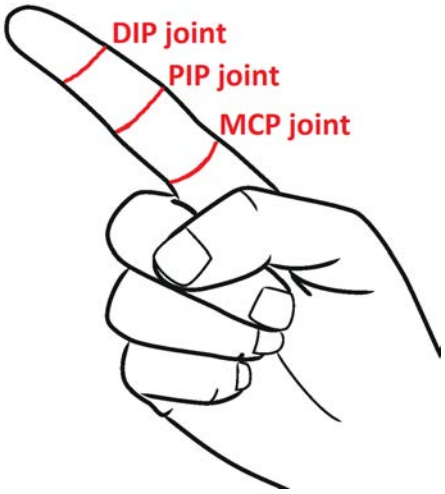


Figure 6.52.: Joint notation of the fingertip close (distal interphalangeal (DIP)) joint, middle (proximal interphalangeal (PIP)) joint, and palm close (metacarpophalangeal (MCP)) joint.

6.5. Ring-based Finger Tracking

In the previous sections, technical properties were studied, the static and dynamic interaction space were determined, and the around-device interaction space was evaluated. This section evaluates whether it is possible to map the capacitive measurements to the finger angles enabling multi-finger tracking. For this purpose, a long short-term memory (LSTM)-based finger tracking algorithm is developed and evaluated.

6.5.1. LSTM-based Finger Tracking

To enable complete tracking of each finger, at least the angles of the fingertip close joint (distal interphalangeal (DIP) joint), middle joint (proximal interphalangeal (PIP) joint), and palm close joint (metacarpophalangeal (MCP) joint) for each finger (Figure 6.52) must be determined. That means the four capacitive sensor readings of PeriSense must be mapped to 15 joint angles. To reduce the complexity, only the PIP and MCP joints are determined since the DIP joint angle can be approximated as shown by Hrabia et al. (Hrabia et al., 2013). Consequently, the angles to be determined are reduced to 10.

Due to the low number of sensors arranged around a ring, the limited sensing field, the non-linearity of capacitive proximity sensing, and the influence of the capacitive measurements by all fingers and the palm, the mapping problem is highly non-linear. These limitations can produce the same or very similar capacitive measurements for different finger poses.

In order to cope with the mapping ratio of low input and high output dimension and the non-linearity of the capacitive measurements, a regression model based on a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is applied to map the four sensor values to the ten joint angles. The LSTM is a recurrent neural network architecture where the cells in the hidden layer are complex units that incorporate an internal memory with different gates for setting, deleting, and reading the memory content. This allows each cell to store received information and compare it to later information. The consecutive input sequence is fed into the LSTM piece by piece, one measurement containing four sensor values at a time. Only the output to the last measurement of the sequence is then transmitted further as a result of the entire sequence. The joint angle cannot jump randomly from one value to another, but they in- or decrease continuous angle by angle. This dependence seems to be appropriate for the internal LSTM structure.

As reference data for this regression problem, a Leap Motion camera[3] was used providing a skeleton model. During the training data collection, finger movements were performed over the Leap Motion camera while wearing a PeriSense prototype. Data from both devices were recorded simultaneously. Further implementation and training details regarding the LSTM network are given in the method section.

6.5.2. Method

This subsection describes the method and setup of this evaluation. It formulates the questions to be answered, the participants, the recording procedure, and finally the analysis procedure.

Question

This evaluation will investigate the following question: Is it possible to map the four capacitive sensing measurements to the finger angles enabling finger tracking? In concrete, which fingers and angles can be tracked, and what are the limits?

Gesture Set

In initial tests, it was discovered that the Leap Motion SDK has difficulties tracking specific finger movements correctly. Consequently, six finger movements, gestures called, were defined: (1) all fingers open and close (fist spread in change), (2) from a fist, thumb, index and middle finger open and close, (3) from spread hand, index finger bends, (4) from a fist, thumb and index finger open and close, (5) from spread hand, ring and little finger bend and (6) from spread hand, middle, ring and little finger bend.

[3]Leap Motion: https://www.leapmotion.com (accessed on: 2019/10/04)

Participants

For the experiment, 17 participants for the evaluation and four additional participants for the hyperparameter optimization were recruited. Both groups were disjunctive from the evaluation in the previous sections. All participants could move their fingers painlessly and unrestrainedly. Overall, both groups together consisted of four female and 17 male participants with different hand sizes (hand length between 16 and $19.8\,\mathrm{cm}$ (μ = $18.26\,\mathrm{cm}$, σ = $1.38\,\mathrm{cm}$), span width of the hand between 18.3 and $23.1\,\mathrm{cm}$ (μ = $20.43\,\mathrm{cm}$, σ = $1.56\,\mathrm{cm}$), index finger length between 6.2 and $7.5\,\mathrm{cm}$ (μ = $7.07\,\mathrm{cm}$, σ = $0.52\,\mathrm{cm}$)) and age range between 24 and 47. Three of the participants were left handed. The participant's circumference of the middle finger was between 56 and $70\,\mathrm{mm}$. To enable evaluation with different finger sizes, three rings were produced with different inner diameters (18, 20, and $22\,\mathrm{mm}$). The outer sizes are kept unchanged and are also identical in regards to technical properties such as signal noise and resolution.

Recording Procedure

To capture the motion of as many as possible fingers, PeriSense was worn on the base segment of the right hand middle finger. Also, the left-handed participants performed the experiment with the right hand and reported no difficulties performing the requested movements since the movements were easy and natural in execution. After introducing the experiment and the possibility of getting familiar with PeriSense and the Leap Motion camera, the participants had to perform a calibration procedure. Within this procedure, the participants were required to move the finger for 20 seconds so that the ring was touched everywhere and had maximum distance everywhere to the adjunct finger to capture the maximum and minimum capacitive values, respectively.

First, each gesture was explained, and the participants exercised them. Afterward, 30 seconds of the gesture execution were recorded while the participants could see the hand model provided by the Leap Motion SDK. The Leap Motion camera frames and the capacitive sensing values from the ring were recorded simultaneously.

Analysis Procedure

First, the Leap Motion frames and the ring data were synchronized regarding the timestamps. Additionally, invalid tracking sequences (e.g. parts without recognized hands by the Leap Motion SDK or hands with low confidence values) were eliminated. Afterward, each finger's PIP and MCP joint angle were extracted from the Leap Motion hand model and normalized to 0 and 1. The capacitive sensing values were normalized to -1 and 1 regarding the calibration values. Finally, all data were merged into one data set.

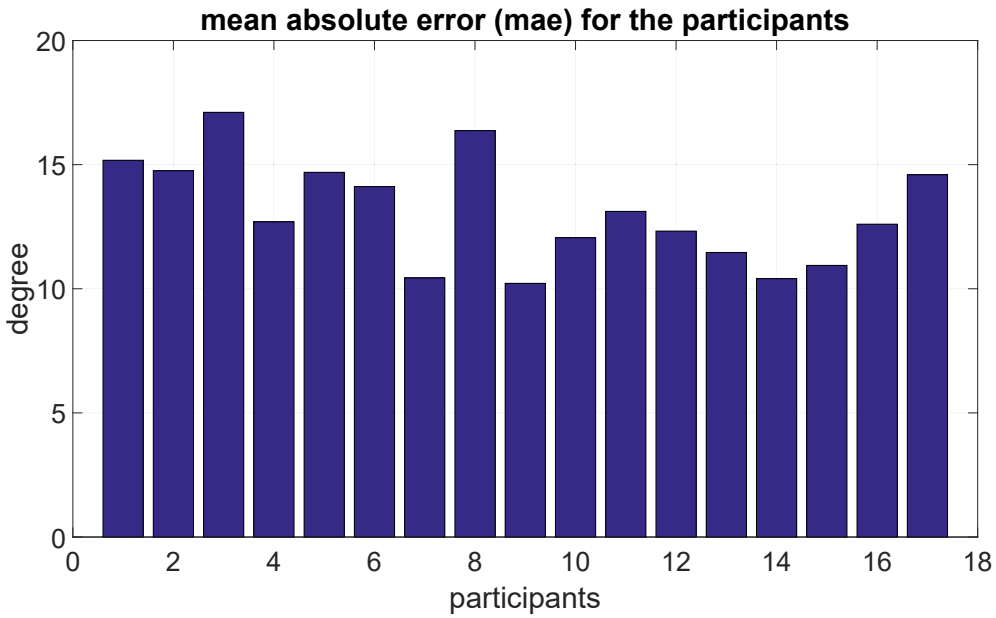


Figure 6.53.: Mean absolute error for each participant.

For both training and evaluation, the loss function is the mean squared error penalizing outliers. Additionally, the absolute error is included as a metric to help interpret the results. The networks are trained using the RMSProp (root mean square propagation) optimizer with a learning rate of 0.001. A sliding window (with a window size of w) was applied to the data set, which generates an (n samples, window size of w, features dimensions) - input matrix for the LSTM layer. The n samples are the number of generated windows, the sliding window size is five, and the dimension of the features (the four capacitive sensing values) is four. All training data (windows) was shuffled before being used. The number of units used in the LSTM cell is 50, and a dropout layer was added with a rate of 0.3 on the LSTM layer. Next, a dense layer with 25 units was added and finally a dense output layer was added with an output dimension of 10 (corresponding to the ten finger angles). The activation function for the two dense layers is a ReLU (rectified linear unit).

The parameter used in the evaluation were determined by a hyperparameter optimization using data from four users, which were not part of the data set used for the evaluation. The data of the 17 participants was used for a leave one user out cross-validation.

6.5.3. Results

The leave one user out cross-validation over the data of 17 participants and all joint angles achieved a mean absolute error (MAE) of 13.02 degrees (minimum MAE is

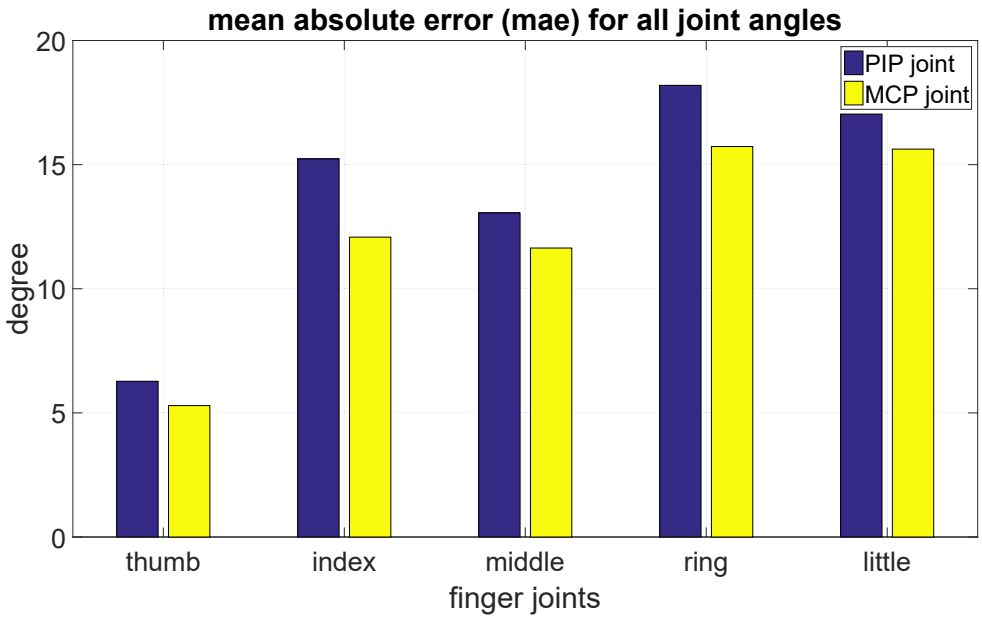


Figure 6.54.: Mean absolute error for each joint.

5.29 degree, maximum MAE is 18.19 degree, and the standard deviation is 4.35). Figure 6.53 shows the MAE for each participant. It reveals that the participants with the largest and smallest hands had the largest error. Participants 3 and 8 had the largest and smallest hands, respectively. Also, participants 1, 2, and 17 had large hands. For 10 participants, the MAE is below 13.5 degrees (minimum MAE is 10.22), and the two largest MAE are 17.11 and 16.37 degrees. The standard deviation over the participant's MAE is 2.21.

Figure 6.54 shows the MAE for each joint. The error of all MCP joints is smaller than the error of the corresponding PIP joints. The lowest errors are achieved for the thumb ($PIP = 6.27$, $MCP = 5.23$) and the highest for the ring finger ($PIP = 18.19$, $MCP = 15.73$) followed by the little finger ($PIP = 17.04$, $MCP = 15.63$). The MAE for the MCP of the index and middle finger are similar with 12.08 and 11.64. The MAE of the PIP joint of the index finger is 15.23 and of the middle finger 13.05.

Figure 6.55 shows the MAE for each joint in 5-degree resolution. The joint of the little and ring finger and the joints of the middle and index fingers show similar behavior. The MAE rises in the second third of the range for the thumb, index, and middle finger. Around 20 to 30 degrees, all joints show lower MAE and smaller MAE distributions. For the thumb, index and middle finger joints, the distribution of the MAE becomes larges when the fingers are bend. Further, all joints show outliers at the upper and lower side of the range and only a few in the middle.

Figure 6.56 shows exemplary snapshots of the gestures (1) to (6) for the partic-

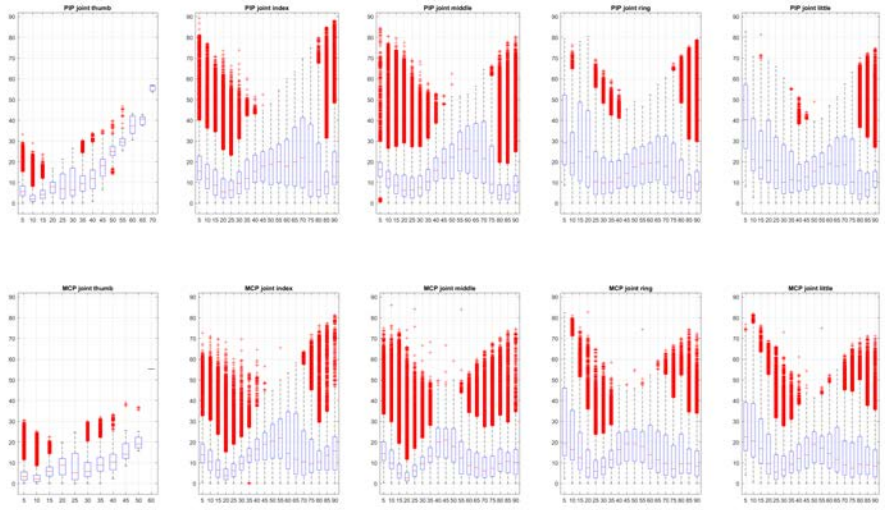


Figure 6.55.: Boxplots showing the MAE distribution over the angle range from 0 to 90 degrees for the each joint.

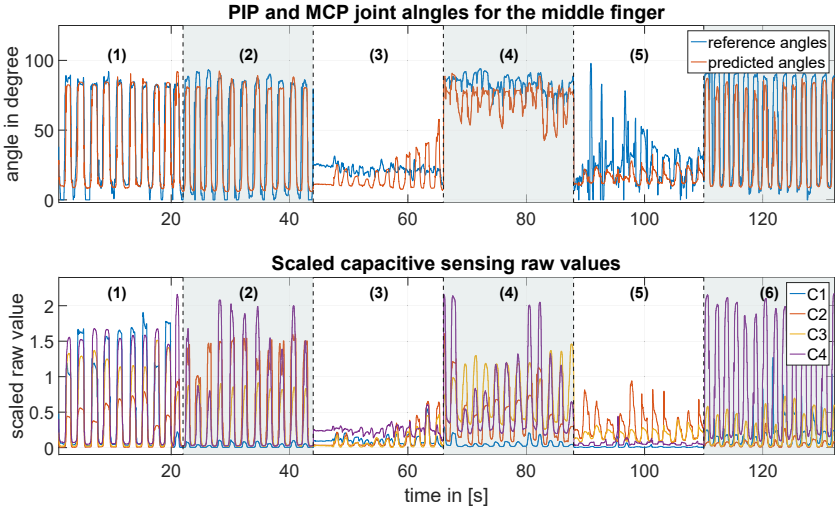


Figure 6.56.: Data snapshots of each gesture ((1) to (6)) from one user. On the top, the PIP joint angle of the middle finger computed by the Leap Motion SDK , and the predicted angle (orange) is shown. Below, the corresponding capacitive sensing values are shown. C1 is the electrode pointing towards the ring finger (to the right side), C2 is at the bottom right and captures the influence of the middle and ring finger, C3 is at the bottom left and measures the influence of the middle and index finger, and C4 points towards the index finger.

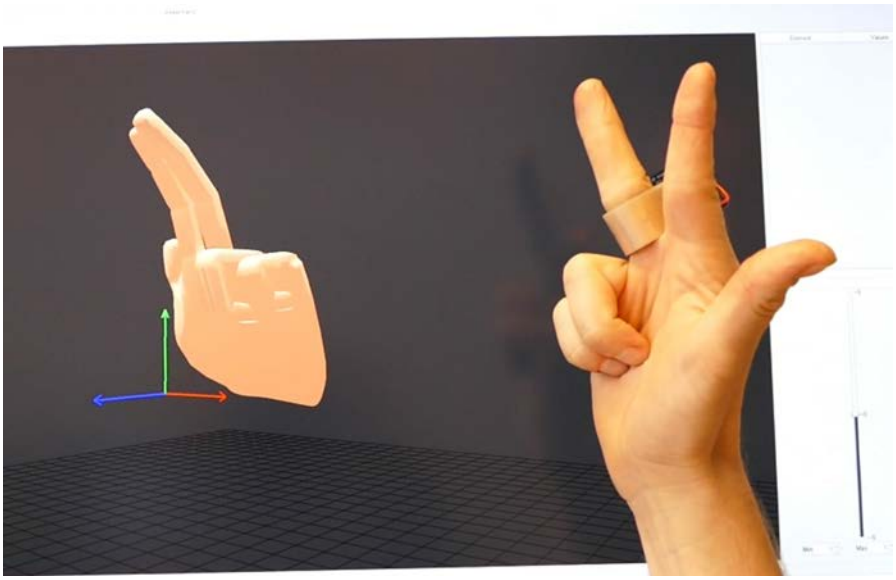


Figure 6.57.: Demonstration setup. It shows the demonstrator's right hand wearing PeriSense on the middle finger and performing finger movements. In the background, there is the (mirrored) computer-animated 3D hand model rendering the demonstrator's finger movements.

ipant number 10's middle finger PIP joint as well as the corresponding capacitive measurements. For gestures (1), (2), and (6), the LSTM tends to underestimate the angle at the upper and lower side of the movement range. Nevertheless, the bending of this joint is predicted regardless if the adjunct fingers moved too or not. At gestures (3), (4), and (5), the middle finger should not move. However, joint bends are predicted up to 50 degrees. On the other hand, the reference angles and the capacitive sensing data show finger bends for gesture (5) where the middle finger should not be moved. In this case, the LSTM predicted less motion than the reference angles.

6.5.4. Discussion

The leave one user out cross-validation results show that very large or small hands can lead to an increased MAE. It is assumed, in such cases, that the ring is not optimal aligned and produces slightly different measurements caused by the different proximities and contacts of the fingers and the palm. The current prototypes scale only inside with respect to the diameter. It would be better also to scale the outer size and form. Further, training different ring sizes separately could enhance performance. Also, user-dependent training should minimize the error.

Although the motion of the thumb and little finger cannot be tracked actively by PeriSense, the thumb shows the lowest MAE. Also, the MAE of the little finger is

comparable with the other ones. The LSTM can predict these angles in dependence on the movement of the other fingers. This is only possible for predefined and natural hand movements like the gestures used in this evaluation (e.g., open and close the hand oscillatory). Using free arbitrary finger movements would cause significant errors for these fingers. Also, gestures consisting only of thumb or little finger movements, such as the shaka or thumb-up gestures, are not detectable because the ring is enclosed by the index, middle, and ring finger. Wearing the ring on the index finger could enable tracking of the thumb, but on the other hand, the ring finger would become not trackable.

The predicted middle finger bends for gestures (3), (4), and (5) (see Figure 6.56) are caused by false tracking of the Leap Motion SDK. This is also visible at gesture (5), where the reference angles show finger bends, although there was no movement. The capacitive measurements show some amplitudes for electrodes C2 and C3, pointing down right and down left, respectively. The movement of the ring finger causes the amplitude of C2. The index finger causes the amplitude of C3. The index touched the electrode slightly when all fingers were stretched. When the ring and little finger are bent, the index finger spread slightly to the left (away from C3). Consequently, the LSTM learned the errors in the reference data, even if the errors do not occur often. Taking more training data could decrease the influence of this error. Alternatively, also better tracking algorithm for the reference system would reduce the errors.

Since it is difficult to imagine the effect of a 13.02 degree error in practice, PeriSense and the finger tracking approach were connected to a virtual 3D hand model (Figure 6.57). In first tests, strong jitters were observed in the virtual model. In consequence, an exponential smoothing with an alpha of 0.5 was applied on the predicted angles, which has been determined by experimental evaluation. This results in a stable visual hand model. According to subjective impression, the angles are displayed correctly, and the fingers move like the real ones. For some movements like gesture (5), the already above-discussed effect of learned errors from the reference data could be observed. Also, movements that are not an explicit part of the training data, like stretching the middle finger, could not be properly depicted by the hand model. Overall, the tracking performance is sufficient enough for displaying natural finger movements. Using a more accurate reference system and additional gestures for the training data could enhance the tracking performance.

7. Applications

In the previous, a concept for recognizing multi-finger gestures with a ring and the prototypical implementation of this concept in the form of PeriSense were presented. Subsequently, PeriSense was comprehensively evaluated analytically. This chapter presents some applications for PeriSense. The fields of application are very manifold. For example, PeriSense can serve as an input device for AR and VR glasses or smartphones (for example, to control the camera shutter or the music player). It can also be used as a remote controller for the smart home. One thing that is not the focus of this work but could also be a use case is activity recognition. Activity recognition could be used to trigger implicit interactions. For example, in a cooking scenario, the ring could detect that the user is stirring for X minutes, and the recipe step is then automatically advanced. In this context, PeriSense could also be used to track fitness activities such as running or swimming, or in the healthcare sector, for example, to detect a fall, document drinking behavior, or report lethargic movements. Also, applications in the context of industry 4.0 are of interest, such as an input controller for a digital workbench (Drews and Weyrich, 1997) or testing novel interaction concepts with a digital twin (Talkhestani and Weyrich, 2020).

In the following, we present four different applications with PeriSense that have already been implemented. Since the interaction pipelines are always very different, but the programming is often complex, but at the same time very similar, a framework called PeriSensePy was developed. This framework allows the creation of interaction pipelines without programming. Accordingly, the PeriSensePy framework is presented first, followed by the four applications.

7.1. PeriSensePy Application Framework

The interaction flow for each application is individual. Figure 7.1 shows a simplified typical pipeline. The sensor data is passed from the device driver to an interaction recognition module. The interaction recognition continuously classifies the sensor

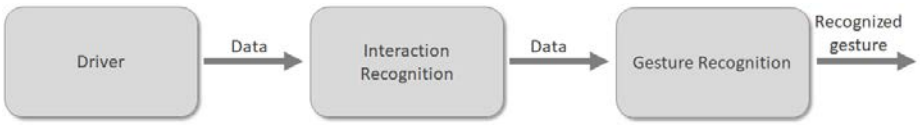


Figure 7.1.: A typical generic interaction pipeline.

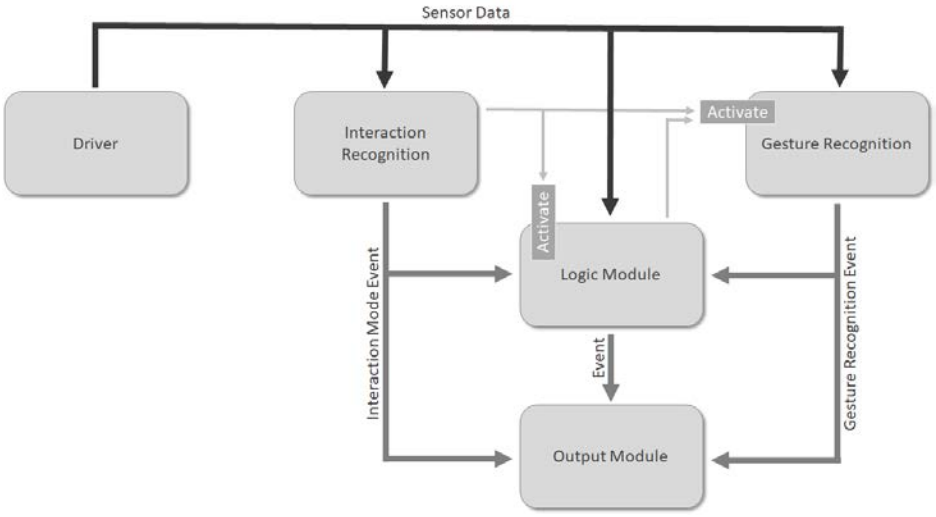


Figure 7.2.: PeriSensePy framework concept.

data into noise (signals that do not contain consciously performed gestures) and gestures. This can be, for example, a gesture filter that classifies, e.g., using a Support Vector Classification (SVC), the signal (such as in Zhang et al. (Zhang et al., 2017a)). However, it can also be an event that characterizes the interaction. For example, the event could be a button press on the interaction device or, in PeriSense's case, tapping the ring with the thumb (if the ring is worn on the index finger). It would also be imaginable that the interaction intention is characterized by a particular hand or finger position or orientation. If the interaction recognition detected an interaction intention, then the gesture recognition is activated and classifies which gesture was performed and whether it is a gesture.

Accordingly, the interaction pipeline must be implemented again for each application. Many code sections are, however, always the same or similar. Nevertheless, they differ considerably in detail, so a simple copy and paste is usually not possible. Due to the complexity of the interaction pipelines' implementation, changes in the interaction flow often require a complete redesign of the implementation.

In order to create interactive applications with PeriSense quickly and easily, a framework called PeriSensePy was developed. It allows the creation of interaction flows without programming. The composition and configuration of the interaction pipelines are done exclusively through a JSON-based configuration file. The interaction pipeline is created and executed from the configuration file at runtime. This allows for quick changes and adjustments without the need to program. Also, the framework allows automatic training of the models defined in the configuration file.

In the following, the implementation concept of the PeriSensePy framework is presented, and afterward, the implementation is described. An example of an interaction pipeline configuration using this framework is presented in Appendix C.

7.1.1. Concept

The goal of PeriSensePy is to provide a generic framework that can be used to implement various and diverse interaction flows. This requires the following modules: a driver module, interaction recognition module, gesture recognition module, logic module, and output module. Figure 7.2 shows all modules and how they are connected and interact with each other.

The driver manages PeriSense and provides the sensor data. The sensor data is independently passed to the interaction recognition, logic, and gesture recognition module. The interaction recognition notifies and activates the logic and the gesture recognition module. The interaction recognition module is always active and continuously processes the sensor data. Optionally, the logic module can also be configured to be always active. The interaction recognition or logic module only activates the gesture recognition module. Deactivation takes place within the gesture recognition module. The interaction recognition module sends an event to the output and logic module. The event may contain more detailed information about the type of interaction recognized, such as the name or a confidence value. The gesture recognition module also sends the recognized gesture as an event to the logic and output module.

The logic module represents a generic component. This module allows the implementation of more complex processes and logic, which cannot be realized via the standard interaction recognition, gesture recognition, and output module. This can be, for example, the implementation of a dimmer by rotating the ring or the fusion of gesture sequences to one command (for example, the fusion of a gesture to address a device followed by a control gesture to a single command).

The output module provides the recognized gestures and events of the interaction recognition and the logic module to applications. This can be, for example, via a web service, REST API, MQTT channel, or WebSocket.

The concept allows the modules to be combined in almost any way. The configuration of the modules is done via a configuration file. The framework evaluates this file at runtime, and the interaction pipeline is created from it.

7.1.2. Implementation

The PeriSensePy framework is implemented in Python. All framework modules have an abstract class, which all concrete module implementations must inherit. The instantiation of the classes is done via corresponding factories. This allows a straightforward and fast extension of the modules by further algorithms.

For the driver module, a PeriSense driver handling the Bluetooth communication and a PeriSense simulation driver is implemented. The simulation driver plays CSV files recorded with PeriSense back into the framework. It thus simulates a PeriSense. Since the data interface is generic, drivers for arbitrary interaction devices

can be added. Thus, the framework is not technically limited to PeriSense.

To implement a concrete interaction detection, the following three functions have to be implemented:

```
def _init(self)

def _classify(self, values)

def train(self)
```

The function *_init(self)* is called when initializing the module implementation and is used for loading the model and other initialization actions. In the function *_classify(self, values)*, the classification of the passed sequence is performed. The function *train(self)* trains the model of the implementation, selects the features, determines the parameters for normalization, and whatever else is necessary for training the specific implementation. Currently, two interaction recognizers have been implemented: TapRecognition and FeatureBasedFilter. TapRecognition performs a tap recognition. It can detect the tap on a specific electrode. FeatureBasedFilter is based on a Support Vector Classification (SVC) from the scikit-learn library[1]. The features are extracted using the tsfel library[2], which are determined via automatic feature selection in the *train(self)* method. The SVC classifies the last n sensor data into noise (i.e., gestures not consciously performed) and gesture.

The gesture recognition module provides the same interface as the interaction recognition module. The following algorithms have been implemented so far: SKLearnRecognition and OneNNRecognition. SKLearnRecognition uses algorithms from the scikit-learn library. There is a choice between SVC, Random Forest Trees, and Multi-layer Perceptron (MLP). OneNNRecognition implements a 1-nearest neighbor classifier. Currently, only Dynamic Time Warping (DTW) is implemented as similarity measure. When using DTW, k prototypes per class can be computed from the training data using Generalized Learning Vector Quantization (GLVQ)(Jain and Schultz, 2018). It will then use the prototypes in place of the original training data for classification. This reduces the number of comparisons in classification. A typical value for the number of prototypes is between 1 and 5 per class, which is much smaller than the total number of training examples.

The logic modules were introduced to implement special cases that cannot be mapped via the other modules. Currently, there is only one implementation of a logic module developed for the smart home controller demonstration in Section 7.2. This implementation determines the users' pointing direction from the motion sensor data and activates the corresponding classifier given the direction.

[1] Machine learning library for Python: https://scikit-learn.org/stable/ (accessed on 10/18/2020)

[2] Time Series Feature Extraction Library (TSFEL for short): https://tsfel.readthedocs.io/en/latest/index.html (accessed on 10/18/2020)

The abstract class of the output module provides the following functions to be implemented:

```
def connect(self)

def disconnect(self)

def _send_message(self, eid, label, name, score)
```

The *connect(self)* function is called during module initialization. Here, a server's connection can be established, a server can be started, or variables can be initialized. The *disconnect(self)* function is called when the interaction pipeline is terminated.The output modules can also be configured via the parameter "*run_forever : false*" so that the two functions are always called when PeriSense connects and disconnects, respectively. The function *_send_message(self, eid, label, name, score)* sends the detected gesture to the corresponding application and if needed also the id of the classifier (*eid*), the label of the gesture (*label*), and the confidence value (*score*). Currently, modules are available for a general MQTT channel, the IOLITE[3] MQTT interaction channel, REST API, WebSocket client and server, output to the console, and a connector for a smart mirror.

The modules can be assembled into an interaction pipeline via an HJSON[4]-based configuration file. In theory, any number of pipelines can be defined and combined per application. The configuration was based on HJSON because it supports comments. This allows fast editing of the configuration file.

A runtime application *ApplicationPipeline.py* is implemented, which creates the pipelines from the configuration and initializes and starts the modules according to the configuration. The training of the individual modules is realized via the *TrainingPipeline.py* runtime application.

The framework contains many more submodules and classes for auxiliary functions, algorithms, data processing, etc. The complete framework description and especially the description of the HJSON file's parameters and the many configuration options are documented in the code repository[5].

An example of an interaction pipeline configuration using this framework is presented in Appendix C. In the following, four example applications using PeriSense are presented.

[3] IOLITE is a smart home platform: https://iolite.de (accessed on 25.03.2021)
[4] HJSON is a user-friendly to read version of JSON:https://hjson.github.io (accessed on 20.03.2021)
[5] The code is placed in the DAI-Labor's git repository: https://gitlab.dai-labor.de/gestureprojects/perisensepy (accessed on 25.03.2021)

Figure 7.3.: Controlling smart home ceiling lights using PeriSense.

7.2. Smart Home Controller

Smart homes contain various networked devices which can be combined to complex home automation scenarios. The user experience can be improved by flexible interaction systems that enable the user to interact with any device from anywhere at home in their daily life. The ability to be worn throughout the day makes PeriSense a suitable interaction device in this context. Frequent automation scenarios, like switching on a light or shutting down all devices when leaving home, could be tied to representative gestures to make them available to the user anywhere throughout their daily routine. Furthermore, the ability to draw unistroke gestures above the ring could be used as textual input, e.g., to enter a pin for door opening.

In the Ambient Assisted Living Testbed of DAI-Labor at the Technische Universität Berlin, a demonstration was developed to test and showcase PeriSense. This demonstration was already presented to a large and broad audience at countless events such as the Long Night of Science[6] or internal events with visitors from science, industry, economics, politics, public sector, and many more. The demonstration consists of three seamless integrated use cases: (1) smart home appliance control, (2) display control, and (3) music center control.

[6] Long Night of Science's website: https://www.langenachtderwissenschaften.de/en (accessed on 24.03.2021)

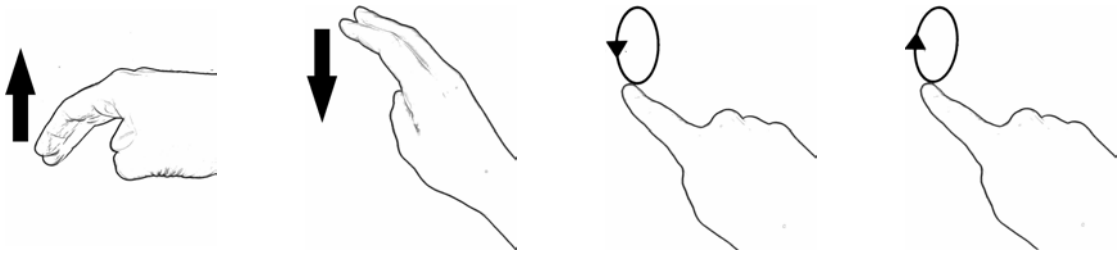
Figure 7.4.: Controlling smart mirror using PeriSense.

7.2.1. Example Scenario

A user wears PeriSense on the right hand's index finger. The user comes into the sleeping room. He points at the ceiling lights with his right index finger and double-taps PeriSense with his thumb. Afterward, he performs a gesture, and the light is turned on (Figure 7.3). He points to the blinds, taps PeriSense again twice, and executes a gesture after which the blinds are lowered. He sits down and points to the music system. He taps PeriSense twice with his thumb and switches on the music center with a gesture. Subsequently, he selects the desired track and volume by gesturing. He picks up his book and starts reading. After a while, the user puts the book aside and turns off the music system by gesturing. He gets up and goes to his smart mirror to find out about tomorrow's weather and appointments. To do this, he points to the mirror and double-taps PeriSense. The mirror displays a visual icon to indicate that it is now active and can be controlled by PeriSense. The user opens the weather forecast tile for tomorrow with a gesture and switches to the calendar tile, performing a gesture afterward (Figure 7.4).

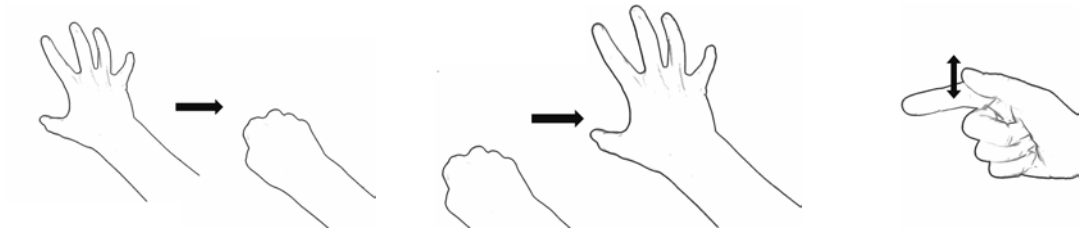
7.2.2. Gesture Sets

The showcase implemented in the Ambient Assisted Living Testbed consists of three use cases. For each use case, a separate gesture set was defined. Figure 7.5 shows the gestures for the device control use case. Gestures Up and Down are used to open and close the blinds. Gestures CircleLeft and CircleRight are used to turn on and off the ceiling lights.



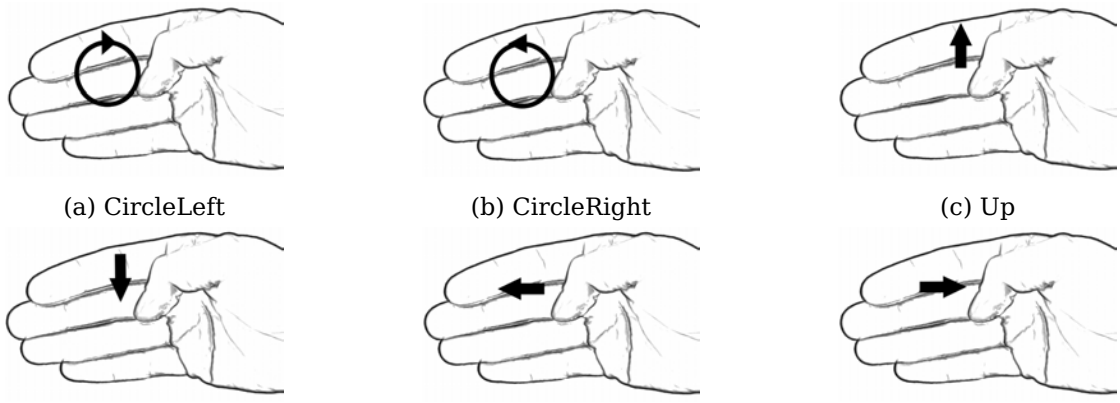
(a) Up (b) Down (c) CircleLeft (d) CircleRight

Figure 7.5.: Smart home device control gesture set.



(a) Grap'nPull (b) Push'nRelease (c) Slide Up/Down

Figure 7.6.: Smart mirror control gesture set.



(a) CircleLeft (b) CircleRight (c) Up

(d) Down (e) Left (f) Right

Figure 7.7.: Smart music center control gesture set.

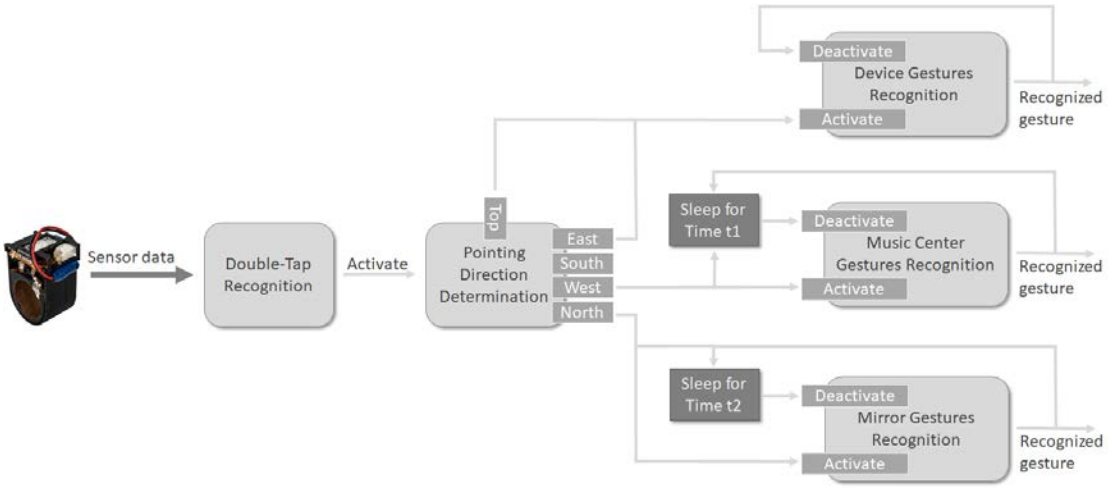


Figure 7.8.: Interaction pipeline for the smart home application.

In the Ambient Assisted Living Testbed, a smart mirror is installed displaying different content. It provides a summary view showing four tiles. Each tile can be enlarged to fullscreen. In the fullscreen mode, it can be switched between the enlarged tiles. Figure 7.6 shows the control gestures for the smart mirror. Gesture Grap'nPull switches from summary view to fullscreen view of the tile in the upper left corner. The gesture is executed by first performing a grasp motion with the hand and stretched limb followed by pulling back the limb. Gesture Push'nRelease switches back from the fullscreen mode to the summary view. It is performed by jabbing the limb and opening the fist. Between the fullscreen tiles can be switched using gestures Left and Right. By sliding the thumb up and down on PeriSense, the tiles are switched.

Figure 7.7 shows the gestures for the music center control use case. The gestures are drawn with the thumb on the palm. Gestures CircleLeft and CircleRight are used to turn on and off the music center. Gestures Up and Down are used to increase and decrease the volume by 10 steps. Gestures Left and Right switch between the music tracks.

7.2.3. Implementation

This demonstration is implemented in the sleeping room of the Ambient Assisted Living Testbed. The device's placement is as follows: the blinds are located eastwards, the ceiling lights at the top, the smart mirror is westwards, and the music center is northwards placed. To ensure that no control command is accidentally triggered by a true-false recognized gesture, the user must double-tap PeriSense with the thumb. The tap recognition is a threshold-based algorithm measuring the change of the capacitive sensing values. The pointing direction is determined by

the compass of the motion sensor. For each use case, a specific gesture recognition algorithm was trained. Figure 7.8 shows the interaction pipeline. The device selection is a logic module. It activates the gesture recognition modules relating to the pointed device. Each use case uses a 1-nearest neighbor classifier for gesture recognition. In the case of the device control, the gesture recognition module deactivates after it recognized a gesture. In the case of the smart mirror and the music center, the gesture recognition is active for $15\,\mathrm{s}$ and $5\,\mathrm{s}$ respectively. Every time a gesture is recognized, the activation time is reset and starts to count from the beginning. This ensures the user has not to tap PeriSense every time if the user wants to turn on the music center and adjust a certain track, e.g.

PeriSense is connected with a Raspberry Pi where the PeriSensePy framework runs. PeriSense connects automatically to the Pi as soon it is turned on. The music center, the blinds, and the ceiling lights are connected and controlled with the IOLITE smart home platform. The recognized control gestures for these devices are sent via an MQTT channel to IOLITE. The smart mirror control interface can be accessed directly via a REST API.

Figure 7.9.: Selecting the focused lights by tapping PeriSense.

Figure 7.10.: Adjusting the brightness by swiping on PeriSense.

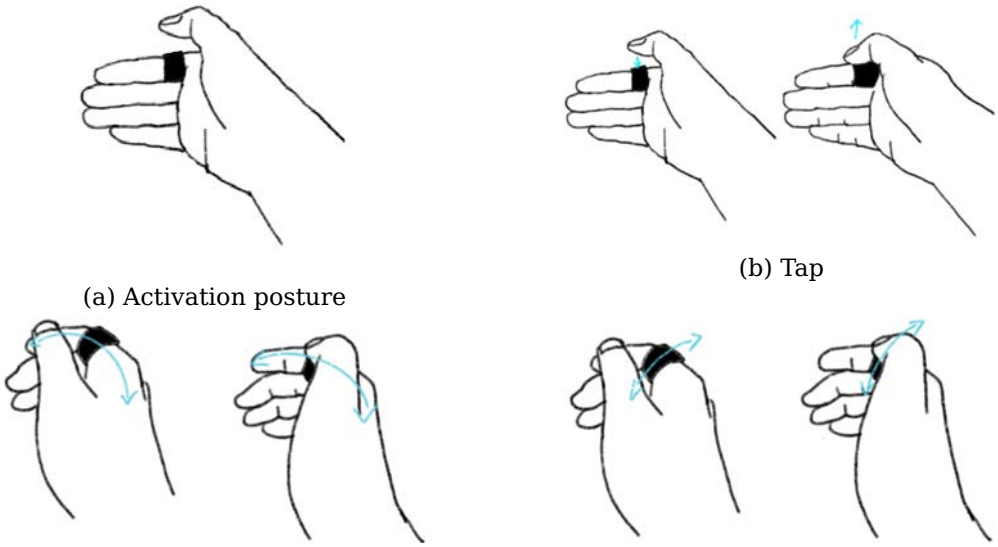
7.3. Input Device for Augmented Reality Glasses

The current trend towards augmented and virtual reality also raises the demand for new input modalities. Existing technologies use camera technology or dedicated input controllers to detect hand input and gestures. PeriSense could achieve similar results but does not suffer from occlusion or light conditions and leaves the user with free hands. The ring could serve to detect directional changes and movements via established motion-sensing techniques. Capacitive sensing can complement this by being able to sense a variety of interactions. In the context of the project UbiAct[7], PeriSense is applied as an input controller for AR glasses in the context of smart home control.

7.3.1. Example Scenario

The user is wearing PeriSense and AR glasses. He wants to start cooking. Therefore, he wants to switch on the light in the kitchen. To do this, he brings his hand into

[7]The UbiAct project was founded by the German Federal Ministry of Education and Research(grant no. 16SV8277): https://www.interaktive-technologien.de/projekte/ubiact (accessed on 17.03.2021)



(a) Activation posture

(b) Tap

(c) Swipe Left/Right

(d) Swipe Up/Down

Figure 7.11.: UbiAct gesture set.

an activation position, which causes all controllable devices in his field of view to be displayed on the AR glasses. The device in the center of the field of view has the focus highlight, in this case, the kitchen lamp (Figure 7.9). He selects the focused kitchen lamp by double-tapping the ring with his thumb. The control menu of the lamp opens. The user can choose between setting the brightness and the color (Figure 7.10). The user can swipe to adjust the lamp's dimming level by looking at the brightness setting. By opening the hand or releasing the activation position, the display on the glasses is closed, and he can start cooking.

7.3.2. Gesture Set

Since the UbiAct project was still running when this thesis was written, the interaction concept presented here is only an early intermediate draft concept. Anyway, the idea of the gesture set is to reduce the number of interactions to a minimum. Further, it is aimed at using micro gestures requiring low cognitive load. Figure 7.11 shows the first sketch of the gesture set. Tapping once PeriSense (Figure 7.11b) selects a focused element. Tapping twice PeriSense (Figure 7.11b) returns to the previous menu or view. Figure 7.11a shows the activation posture.

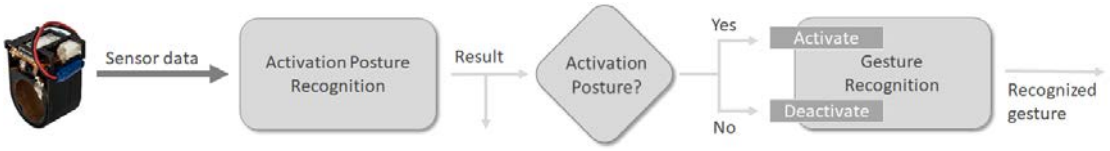


Figure 7.12.: Interaction pipeline for the UbiAct use case.

7.3.3. Implementation

UbiAct's current concept is to provide PeriSense with the IOLITE platform[8]. Accordingly, PeriSense is connected to a Raspberry Pi running the PeriSensePy framework and the IOLITE platform. Figure 7.12 shows the interaction pipeline. The gesture recognition module is active once the user's activation posture is detected. The detected gestures and de-/activation events are sent to the IOLITE platform over an MQTT channel. The platform then pushes the gestures to the AR glasses, which are also connected to the platform via a REST API.

[8]IOLITE smart home platform: https://iolite.de/en (accessed on 24.03.2021)

7.4. Drone Controller

Special Operations Forces (SOF) are facing extreme risks when prosecuting crimes in uncharted environments like buildings. Autonomous drones could potentially save officers' lives by assisting in those exploration tasks, but an intuitive and reliable way of communicating with autonomous systems is yet to be established. In the context of the project InLaSeD[9], an interaction concept using PeriSense was designed to be used by SOF during operation for interaction with autonomous systems (Montebaur et al., 2020). Controlling autonomous drones using gestures is more intuitive than using a remote control. Additionally, using PeriSense, there is no need to hold a device in the hands and look at it producing additional cognitive load and attention.

7.4.1. Example Scenario

A SOF unit entering a building is accompanied by an autonomous operating drone delivering visual information to the unit. The leading officer of the SOF unit carries PeriSense and a mobile computer. PeriSense is connected to the mobile computer, where the sensor data is processed and recognized commando events sent to the drone. During the SOF units moves thru the building, the autonomous operating drone supports the unit by inspecting the environment for potential dangers. As the unit approaches another room entrance, the leading officer sends the drone into this room. To do that, he performs a specific commando gesture related to the drone. The drone takes off. Afterward, the leading officer performs another gesture to send the drone forwards into the room, where the drone starts to explore the room. As the room seems to be safe, the leading officer calls the drone back using a gesture. Afterward, the leading officer performs a gesture signalizing two unit members to move forwards into the room.

7.4.2. Gesture Set

Communicating with gestures is already well-known to special forces. Building on this well-established interaction model, additional gestures specifically designed to instruct autonomous drones were designed, assisting police officers during the operation. These defined gestures (Figure 7.13) can co-exist with already established commandos and can not be mistaken for everyday finger and hand movements.

[9]The InLaSeD project was funded by the German Federal Ministry of Education and Research (grant no. 13N14795): https://www.sifo.de/de/bewilligte-projekte-aus-der-bekanntmachung-anwender-innovativ-forschung-fuer-die-zivile-2256.html(accessed on 27.03.2021)

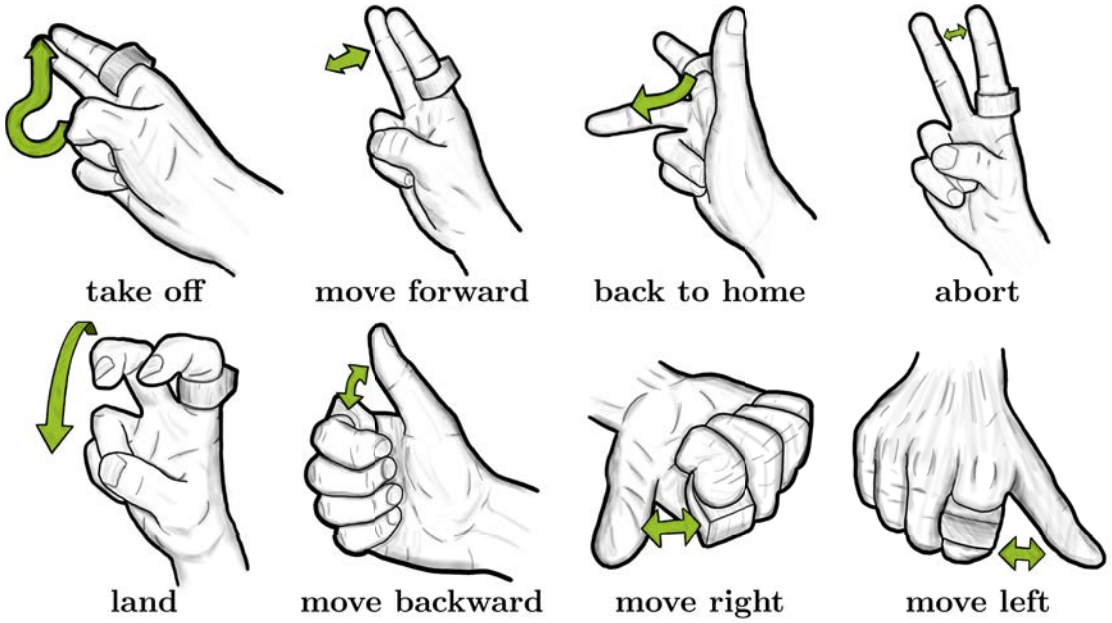


Figure 7.13.: Autonomous drone gesture command set.

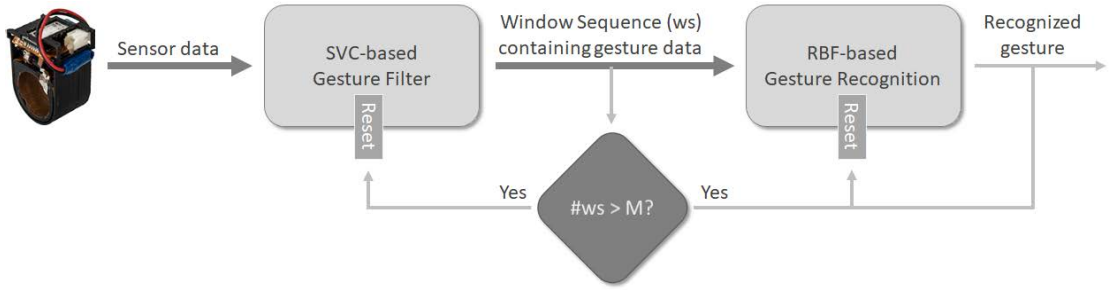


Figure 7.14.: Interaction pipeline for the drone control application.

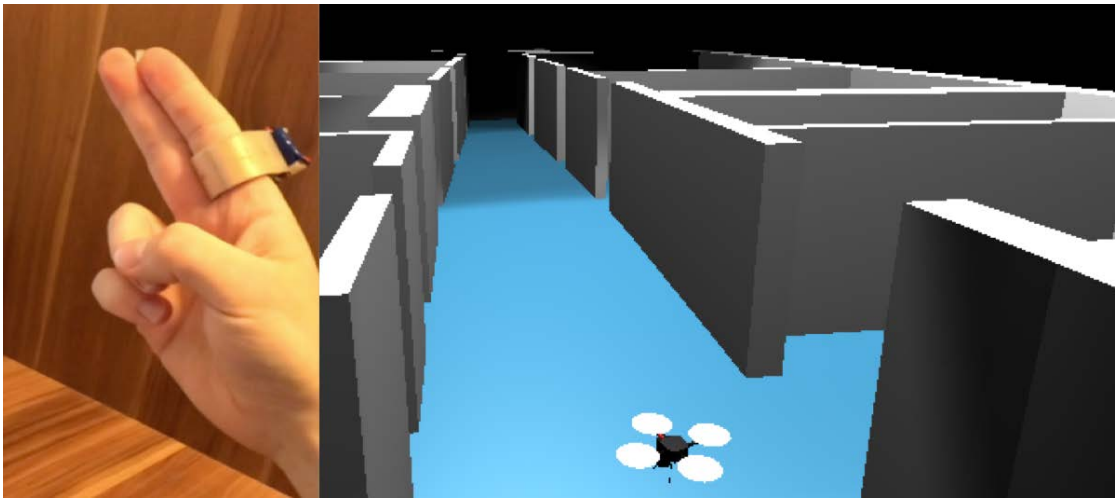


Figure 7.15.: Demonstration setup.

7.4.3. Implementation

This demonstration is prototypically implemented as a simulation. PeriSense is connected with a computer where the PeriSensePy framework is running. Figure 7.14 shows the interaction pipeline for this demonstration. The gesture filter is based on an SVC classifier deciding if the last n sensor values belong to a gesture or not. If these n sensor values are considered as part of a gesture, they are forwarded to a Random Forest-based gesture recognition module classifying the incoming values. If the filter passes m windows (with $m >> averagegesturelength$) in sequence to the gesture recognition, then the gesture recognition and the gesture filter are reset. The gesture filter forwards only a sensor window again when at least one sequence was classified as noise (no indented gesture). This reduces True-False recognition.

The recognized control commands represented by the corresponding gestures are passed to drone simulation based on MORSE[10] running on the notebook computer, too (Figure 7.15). Based on the control command, the simulated drone starts actions like taking off or moving forward.

[10] http://morse-simulator.github.io/ (accessed on 18.12.2020)

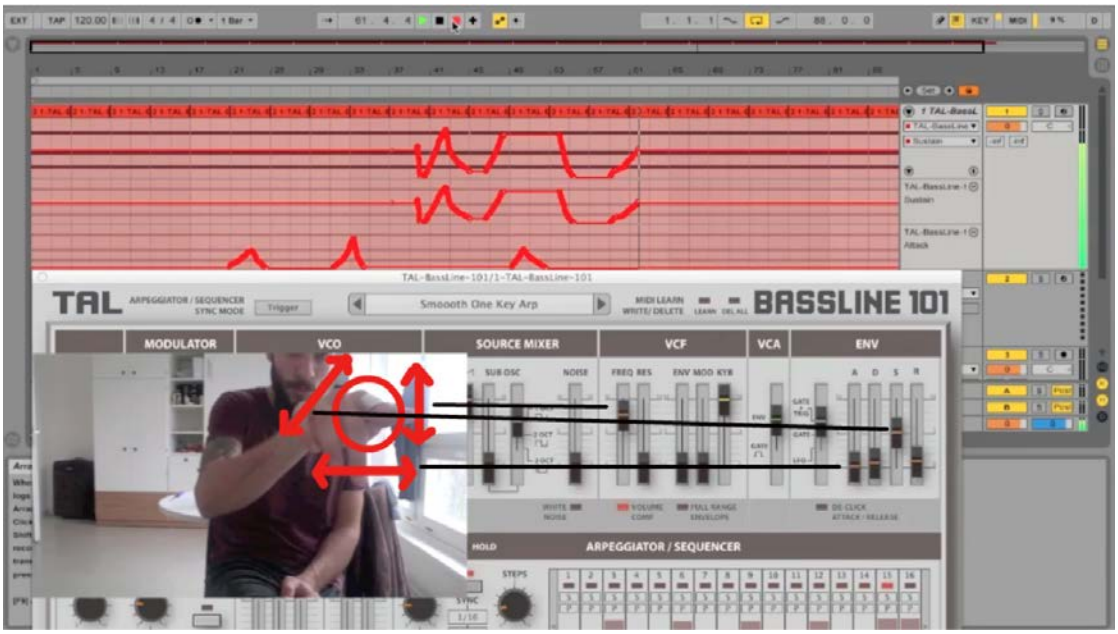


Figure 7.16.: Modulating "Bassline 101" parameters with PeriSense.

7.5. Immersive Midi Controller

The most important aspect for music producers is the fusion with their music to explain their histories, feelings, and ideas. This requires immersive equipment to mix their movement with music in order to get a unique experience. Nowadays, electronic music equipment only has buttons, knobs, and faders to modulate the sounds. This does not create immersion and feels unnatural to musicians. The control with a knob or fader allows only a one-dimensional expression. Consequently, it is becoming more common that electronic music equipment includes touch interfaces in their devices because it provides a better experience.

In the context of Bryan Balfagón's Bachelor thesis (Balfagón Ortiga, 2016), it was studied if PeriSense is suitable to be a natural and immersive controller for musical midi software. In the study with 18 musicians ranging from amateurs to professionals, it was found that using PeriSense as an input controller for midi software takes the creation of music to another level. This allows a new synchronization level between human movement and sound, resulting in a higher artistic potential.

7.5.1. Scenario

In a live show, the musician plays the notes of the main melody or arpeggio with one hand. The melody is the most modulated in electronic music. The rest of the sounds are usually cyclical. With the other hand wearing PeriSense, the musician changes

the MIDI software parameters to modulate the melody. The parameters are mapped to modulate the frequency filter and the ADSR2 module, such as the most common attack and sustain. By moving the hand, the musician can express his emotions in movements, and the movements in turns are translated into the modulation of the melody.

7.5.2. Interaction Concept

Up to five control elements to be modulated can be assigned to each electrode. The control elements to be modulated can be assigned to the directions of movement in the x, y, and z directions and the axes of rotation around the motion sensor's x- and y-axis. With four electrodes, this results in up to 20 assignable control elements. The control elements of an electrode can be activated by tapping on the corresponding electrode. As soon as the electrode has been activated, the movements of the hand are mapped to the corresponding control elements according to the direction of movement. Tapping the electrode again deactivates the mapping. If all four electrodes are activated, up to 20 control elements can be modulated simultaneously. Figure 7.16 shows an example of control assignments for an electrode.

7.5.3. Implementation

PeriSense is connected to a computer with MIDI software installed. The motion data changes of PeriSense are mapped to controller values of the MIDI software. The mapped controller values are decoded into MIDI messages and sent to the MIDI software via the MIDI port. The mapping of the sensor values to the corresponding controller values is configured via a user interface. The entire demo is implemented in Java since the PeriSensePy framework did not exist at that time. However, it is possible to implement the interaction pipeline with the PeriSensePy framework. Figure 7.16 shows the demo. It shows a MIDI software in the background and a camera snapshot on the left-bottom, which shows a user wearing PeriSense. Also, the mapping between the sensors and the controller is drawn.

8. Conclusions

This dissertation aims to evaluate the capabilities and limitations of capacitive sensing for multi-finger interaction with a ring. For this purpose, a ring-based interaction device using capacitive sensing for multi-finger interaction was proposed, and a prototype, PeriSense called, was developed. Based on PeriSense, comprehensive evaluations were conducted. Besides the technical properties, the static and dynamic interaction spaces were determined. Additionally, the around-device interaction space was also evaluated. Subsequently, a finger tracking algorithm mapping the capacitive sensor readings to finger angles was developed and evaluated. Finally, four different interactive applications using PeriSense were presented. To implement these applications, a framework was developed. This framework allows creating complex interaction pipelines without the need for programming via a JSON-based configuration file.

This chapter summarizes and discusses the contributions and evaluation results with regard to the research questions defined in Section 2.2. Afterward, PeriSense's limitations and the resulting future work are discussed.

8.1. Summary

This section summarizes and discusses the concept and the evaluation results concerning the research questions defined in Section 2.2. This thesis first proposed a concept for a ring enabling multi-finger interaction. For this purpose, the ring was equipped with four electrodes measuring the capacitance between nearby fingers or the hand palm and the corresponding electrode. Finger movement changes the distance between the fingers and the electrodes, causing a change in the measured capacitance. This effect is used to distinguish between different finger movements and interactions in general.

Based on this concept, a prototype, PeriSense called, was developed. PeriSense consists of a 2-layer flexible circuit board layout. For the capacitive sensing measurement, a Texas Instruments FDC1004 sensor (TexasInstruments, 2015) is used. An additional inertial measurement unit (IMU) with 9 degrees of freedom is equipped to augment sensor readings for gesture recognition with hand position and orientation tracking. A microcontroller gathers the sensor data and transmits them via a Bluetooth module to a (mobile) computer for further data processing. The power consumption of the hardware without the Bluetooth module is about $50\,\mathrm{mW}$ and

with the Bluetooth module enabled at about $380\,\mathrm{mW}$. Equipped with a $130\,\mathrm{mA\,h}$ battery, PeriSense's runtime is about $1\,\mathrm{h}$ to $1.5\,\mathrm{h}$. The flexible circuit board allows for a convenient way of placing all electrical components and a battery in a 3D-printed finger ring casing. PeriSense is thus a self-containing prototype that can be used for evaluations and demonstrations.

Three PeriSense versions with different inner diameters but the same outer diameter and technical properties were built and used for evaluations. In the following, the evaluation results are discussed concerning the research question.

Question 1: *Which technical properties can be achieved using standard consumer electronic components?* The technical evaluation has shown that ambient noise does not influence the measurement. The ring should only be used for interactions where the ring is worn on the finger so that there is a reference mass reducing noise. Direct contact with devices containing radio modules or generally high electrical emissions may occasionally result in increased noise in the signal. In practice, however, this should not be a limitation.

The measurement of the resolution showed that the effective range reaches up to $2.5\,\mathrm{cm}$. Consequently, at least the influence of the neighboring fingers can be detected. Only the thumb is borderline in detection when the ring is worn on the index finger. This is reasoned because the thumb is directed approximately 90 degrees to the index finger when fully spread. The field is directed so that it does not perceive lateral influence. Larger electrodes could increase the measurement range. However, this also increases the ring's form factor reducing the wearing comfort.

Question 2: *Which gestures and interactions are possible and where are the limits?* To answer this questions, three sub-questions were derived and answered first.

Question 2-1: *What is the static interaction space?* The ASL fingerspelling dataset was used for evaluation to test which types of postures are recognizable. This dataset consists of many different finger poses involving all fingers. In the case of user and session-dependent cross-validation, a precision up to 0.78 could be achieved. Regarding the interaction space, it can be concluded that the thumb is difficult to measure unless it is close to or touching the electrodes. This results from the directed electric field, which is more orthogonal to the electrodes. The thumb, however, usually moves below the ring and thus the electric field. This results in high confusion between the pinch gestures, such as for the numbers 6 to 9. Also, the pinky finger's posture is difficult to detect because it is slightly out of range or often covered by other fingers.

The two main issues that could be observed are the ring prototype's width and variations in executing the calibration posture. The ring already had a little bit of contact with the skin varying with different finger postures for users with shorter fingers. Therefore, a flexible circuit board design scaling with inner ring size would be advantageous. In this case, the outer diameter could also decrease with a smaller

inner diameter resulting in a smaller ring casing.

Also, the calibration process seems not to be optimal chosen. For example, slight variations in the calibration posture's execution, such as over-stretching of the index finger or bending it too much, result in biased calibration sequences. Also, hand and finger forms, as well as the finger flexibility, lead to a bias in the calibration. This bias can cause a change in the ratio between the four signal baselines, making it difficult to postures from different users.

In summary, postures can cause difficulties in the recognition since the signal offset depends on the user's hand and body ground making an offset-elimination difficult. Also, slight changes in the ring orientations during wearing must be considered. Nevertheless, optimizing the offset handling and feature selection, a user-specific classifier could be trained on the ASL dataset A to Z (without numbers), achieving good results. Despite this experiment aimed to investigate the interaction space during static finger positions, the dynamic part of the posture, i.e., the change or finger movement between letter transitions, could also be added for classification. This would contribute to an improvement in classification performance.

Question 2-2: *What is the dynamic interaction space?* To evaluate the dynamic interaction space, various gestures different in execution but partially similar to the resulting sensor pattern were selected. In the case of user and session-dependent cross-validation, a precision up to 0.98 and on average 0.93 could be achieved. The results dropped in the case of ring repositioning and the user-independence test. The main confusion was between Circle and Rectangle. These gestures were challenging to perform for the users because they were only allowed to move the finger without moving the hand. This resulted in strongly varying patterns between the users.

In general, it can be concluded regarding the interaction space that PeriSense produces proper distinguishable patterns for finger gestures based on capacitive measurements. However, similar motion and variations in execution can lead to confusions. Furthermore, it is to assume that smaller and more electrodes could also reduce confusions. The relatively large electrodes seem to hinder the sensing of smaller finger displacements along the electrodes.

Question 2-3: *What is the around-device interaction space?* To evaluate the around-device interaction space, small unistroke gestures drawn with the other hand's index finger above the ring were used for the evaluation. The test revealed that PeriSense could also detect around-device interactions performed with the other hand, in this concrete case, two-handed unistroke gesture input. In general, this input technique generates well distinguishable patterns for most gestures. The big confusion group of Triangle, Rectangle, Circle, and Delete seems to arise from the one-dimensional electrode arrangement since it is impossible to detect the direction of movement along the electrode.

There is no significant difference between user dependence and user indepen-

dence tests. Changes in the ring orientation are no issue since the ring is fixed between the index and middle finger. Only for one user out of ten, a low result was obtained. Judging from the signals, in this case, it seems that the gestures were not performed exactly over the ring.

The ability to detect multiple finger gestures and around-device interactions increases the interaction space and, thus, the variations of applications. Further detectable around-device interaction would be palm gestures as used in (Zhang et al., 2017a) for example. Here, the ring could be worn at the index finger, middle finger, or thumb.

Summarizing the results regarding **Question 2** overall, a wide variety of around-device interactions and multi-finger movements can be detected and distinguished. Only postures can be problematic because of the varying signal offset. Also, user-independent recognition and ring repositioning can be problematic. However, increasing the number of electrodes and reducing the ring size could meliorate these drawbacks and improve the recognition for all interactions.

Question 3: *Can the interaction space increased by additional sensor data information?* For the posture and gesture interactions, the additional motion sensor data improved the recognition results. In particular, it helped to reduce confusions caused by repositioning of the ring and variations between users. Additionally, the motion sensor extends the interaction space by hand and am movements and orientations.

In general, it is possible to integrate additional sensors into PeriSense. For example, a camera or a piezo microphone can provide context information or even increase detection accuracy and resolve confusions. However, each additional sensor increases power consumption and significantly reduces battery life, threatening PeriSense's self-sufficiency. High-performance batteries suitable to be installed in a ring casing are still the subject of research. Therefore, the usage of capacitive sensing and a motion sensor represents a perfect compromise regarding the resulting interaction space.

Question 4: *Is it possible to map the ring data to a hand model in order to determine the angles of each finger joint?* To enable complete tracking of each finger, at least the angles of the fingertip close joint (DIP joint), middle joint (PIP joint), and palm close joint (MCP joint) for each finger must be determined (Figure 6.52). That means the four capacitive sensor readings of PeriSense must be mapped to 15 joint angles. To reduce the complexity, only the PIP and MCP joints are determined because the DIP joint angle can be approximated, as shown by Hrabia et al. (Hrabia et al., 2013). Consequently, the angles to be determined are reduced to ten.

In order to cope with the mapping ratio of low input and high output dimension and the non-linearity of the capacitive measurements, a regression model based on a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) was applied to map the four sensor readings to the ten joint angles. The evaluation showed

that the average mean error over all angles is only 13.02 degrees. At least the motion of three fingers can be tracked reliable, and in the case of predefined natural movements, even all five fingers' angles can be estimated. As a reference system for learning the mapping, a Leap Motion camera and the corresponding SDK were used. Unfortunately, the finger tracking of the Leap Motion SDK is too inaccurate. These tracking errors are trained by the LSTM, which in turn leads to inaccurate tracking with PeriSense. A more accurate reference system would significantly improve the mapping model, leading to a minor mean average error. Additionally, like postures, finger tracking is also affected by variations in offset. A smaller ring size to eliminate unwanted finger contacts would also reduce the tracking error.

The extensive evaluations revealed promising results. However, the experiments and gesture sets were of experimental nature and with little relation to applications. Therefore, four application examples were presented, which have already been implemented and are in use. Here, PeriSense's wide range of possible applications is evident due to its broad interaction space. The applications include smart home control, display control, drone control, and input device for MIDI applications. Since implementing the specific interaction flows is always time-consuming and turns out to be complex, a framework has been developed. The framework allows the creation of complete interaction recognition pipelines without programming via a JSON-based configuration file. This allows fast and flexible creation and modification of interaction flows.

8.2. Future Work

The evaluations indicated PeriSense's ability to distinguish a broad range of gestures and provide various interaction techniques. However, some limitations were disclosed as well. Initially, it was assumed large electrodes with higher sensing resolution at larger distances would benefit gesture recognition accuracy. However, it is estimated that decreasing the electrode size while increasing the number of electrodes can be advantageous for most gestures. Currently, the electrodes are arranged in one line across the ring's outer surface. Arranging them in two dimensions on this surface could lead to noticeable increases in sensor performance, particularly for around-device interactions. Smaller electrodes additionally enable smaller ring forms. This would foster wearing comfort and reduce unwanted finger contacts inducing an additional offset. However, a trade-off between electrode size and its sensor range has to be taken.

Regarding the ring size, a flexible circuit board design scaling with inner ring size would be advantageous. In this case, the outer diameter could also decrease with a smaller inner diameter resulting in a smaller ring casing. This would increase the wearing comfort and reduce unwanted finger contacts.

With an increased number of electrodes, it could be possible to normalize the sen-

sor data regarding ring orientation along the finger axis. The current electrode size is too large to detect rotational changes. For this normalization, a singular value decomposition (SVD) based algorithm (Horn et al., 1988) could be applied. This normalization would resolve errors caused by repositioning and change in orientation during wearing.

A technical limitation is currently the power consumption of PeriSense. This is mainly founded in the used Bluetooth module consuming about 85% of the power usage. The power consumption of our prototype without the Bluetooth module is about $50\,\mathrm{mW}$ and with the Bluetooth module enabled at about $380\,\mathrm{mW}$. Utilizing a single controller for I2C and RF communication will lead to a more energy-friendly design. Furthermore, additional power-saving routines could be implemented. For example, the low power wake on motion interrupt of the motion sensor could suspend and wake up the microcontroller and its attached sensors. Further, it is also inefficient to send all data points continuously to an external device. Therefore, a low-cost pre-filtering algorithm could help to avoid sending noise and meaningless data and would thereby reduce the number of transferred packages and the overall power consumption of the communication module (Lackerschmid, 2018, e.g.).

When the hardware was designed, Texas Instrument was the only manufacturer to offer an integrated chip for capacitive sensing suitable for PeriSense. In the meantime, there are comparable chips from Analog Devices[1]. These have partly different characteristics and can operate up to 24 electrodes depending on the chip. Therefore, it would be a worthwhile undertaking to compare the chip versions from Texas Instruments and Analog Devices in the application with PeriSense. In this context, a comparison between the loading and shunt mode for measuring the capacitance would also be interesting. The Texas Instruments chip used made it difficult to use the shunt mode due to the limited number of channels available. The chips from Analog Devices have up to 24 channels, which would allow an electrode design for the shunt mode to be implemented for PeriSense.

Regarding finger tracking, it is worthwhile to use a more accurate reference system for learning the images. This would significantly improve the already good result and add value.

The PeriSensePy framework offers an elegant way to implement a variety of interaction flows. The flows' configuration does not require any programming knowledge but expert knowledge about machine gesture recognition. In order to eliminate this expert knowledge and thus make the framework available to application developers or even interaction designers, for example, it requires further algorithms to automate the training and selection of hyper-parameters. Since the framework is generic and modular, it can also be used for other interaction devices. For this purpose, only the driver and data interface needs to be further generalized. In

[1] Analog Device's capacitive sensing chips: https://www.analog.com/en/products/analog-to-digital-converters/integrated-special-purpose-ad-converters/capacitive-to-digital-and-touch-screen-controllers.html (accessed on 30.03.2021)

addition, as the number of interaction devices increases, more gesture recognition algorithms may be needed, such as image processing algorithms. Since the framework is implemented in Python using the multiprocessing library, there are synchronization-related latencies in the data transfer between the threads. High data rates or interaction pipelines with multiple gesture recognition modules can significantly delay recognition under certain circumstances. This can only be remedied by using cython[2], e.g., or re-implementing the framework in another language, such as C++ or Java.

Finally, four implemented example implementations showcasing PeriSense's capability of providing a broad interaction space were presented. However, these applications are still limited regarding real-life scenarios. In the next step, PeriSense has to be tested and evaluated in the wild. This is where the UbiAct project funded by the German Federal Ministry of Education and Research connects. In the context of UbiAct, PeriSense is evaluated as an interaction device for smart home control and as an input device for augmented reality glasses in seamlessly integrated use cases in a real-world environment.

8.3. Conclusion

This dissertation presented PeriSense, a wearable interaction device in the shape of a ring for interaction in the context of IoT. PeriSense is entirely self-sufficient and can be unobtrusive worn the whole day without disturbing the user in his daily routines. By using capacitive sensing and a motion sensor, a large interaction space is enabled. Furthermore, with the PeriSensePy framework enabling the fast and flexible creation of interactive use cases, this dissertation contributes towards an unobtrusive interaction device available throughout the daily routine.

[2] Cython is an optimising static compiler for Python and cPython: https://cython.org (accessed on 23.03.2021)

References

Ahmad, F. and Musilek, P. (2006). UbiHand: A Wearable Input Device for 3D Interaction. In *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH '06, New York, NY, USA. ACM.

Amma, C., Krings, T., Böer, J., and Schultz, T. (2015). Advancing Muscle-Computer Interfaces with High-Density Electromyography. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 929–938, New York, NY, USA. ACM.

Ashbrook, D., Baudisch, P., and White, S. (2011). Nenya: Subtle and Eyes-free Mobile Input with a Magnetically-tracked Finger Ring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2043–2046, New York, NY, USA. ACM.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2017). The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660.

Balfagón Ortiga, B. (2016). *Supporting Musical Midi Software Through Gesture Interaction for a Natural and Immersive to Use Interface*. Bachelor thesis, Technische Universität Berlin, Berlin.

Baxter, L. (1996). *Capacitive Sensors: Design and Applications*. IEEE Press Series on Electronics Technology. John Wiley & Sons.

Bello, H., Zhou, B., Suh, S., and Lukowicz, P. (2021). MoCapaci: Posture and gesture detection in loose garments using textile cables as capacitive antennas. In *2021 International Symposium on Wearable Computers*, pages 78–83. Association for Computing Machinery, New York, NY, USA.

Bian, S. and Lukowicz, P. (2021). Capacitive Sensing Based On-board Hand Gesture Recognition with TinyML. In *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*, UbiComp '21, pages 4–5, New York, NY, USA. Association for Computing Machinery.

Boldu, R., Dancu, A., Matthies, D. J., Buddhika, T., Siriwardhana, S., and Nanayakkara, S. (2018a). FingerReader2.0: Designing and Evaluating a Wear-

able Finger-Worn Camera to Assist People with Visual Impairments while Shopping. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):94:1–94:19.

Boldu, R., Dancu, A., Matthies, D. J., Cascón, P. G., Ransir, S., and Nanayakkara, S. (2018b). Thumb-In-Motion: Evaluating Thumb-to-Ring Microgestures for Athletic Activity. In *Proceedings of the Symposium on Spatial User Interaction*, SUI '18, pages 150–157, Berlin, Germany. Association for Computing Machinery.

Braun, A., Wichert, R., Kuijper, A., and Fellner, D. W. (2015). Capacitive Proximity Sensing in Smart Environments. *Journal of Ambient Intelligence and Smart Environments*, 7(4):483–510.

Buddhika, T., Zhang, H., Chan, S. W. T., Dissanayake, V., Nanayakkara, S., and Zimmermann, R. (2019). fSense: Unlocking the Dimension of Force for Gestural Interactions using Smartwatch PPG Sensor. In *Proceedings of the 10th Augmented Human International Conference 2019*, AH2019, pages 1–5, New York, NY, USA. Association for Computing Machinery.

Chan, L., Chen, Y.-L., Hsieh, C.-H., Liang, R.-H., and Chen, B.-Y. (2015). CyclopsRing: Enabling Whole-Hand and Context-Aware Interactions Through a Fisheye Ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pages 549–556, New York, NY, USA. ACM.

Chatain, J., Sisserman, D. M., Reichardt, L., Fayolle, V., Kapur, M., Sumner, R. W., Zünd, F., and Bermano, A. H. (2020). DigiGlo: Exploring the Palm as an Input and Display Mechanism through Digital Gloves. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '20, pages 374–385, New York, NY, USA. Association for Computing Machinery.

Chen, D., Wang, M., He, C., Luo, Q., Iravantchi, Y., Sample, A., Shin, K. G., and Wang, X. (2021). MagX: Wearable, untethered hands tracking with passive magnets. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, pages 269–282, New York, NY, USA. Association for Computing Machinery.

Chen, K.-Y., Lyons, K., White, S., and Patel, S. (2013). uTrack: 3D Input Using Two Magnetic Sensors. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 237–244, New York, NY, USA. ACM.

Cheng, J., Amft, O., Bahle, G., and Lukowicz, P. (2013). Designing Sensitive Wearable Capacitive Sensors for Activity Recognition. *IEEE Sensors Journal*, 13(10):3935–3947.

Cheshire, S. and Baker, M. (1997). Consistent overhead byte stuffing. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '97, pages 209–220, New York, NY, USA. Association for Computing Machinery.

Chung, J., Oh, C., Park, S., and Suh, B. (2018). PairRing: A Ring-Shaped Rotatable Smartwatch Controller. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, pages 1–6, New York, NY, USA. Association for Computing Machinery.

Dai, Q., Li, X., Geng, W., Jin, W., and Liang, X. (2021). CAPG-MYO: A Muscle-Computer Interface Supporting User-defined Gesture Recognition. In *The 2021 9th International Conference on Computer and Communications Management*, pages 52–58. Association for Computing Machinery, New York, NY, USA.

Dementyev, A. and Paradiso, J. A. (2014). WristFlex: Low-power Gesture Input with Wrist-worn Pressure Sensors. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 161–166, New York, NY, USA. ACM.

Dietz, P. and Leigh, D. (2001). DiamondTouch: A multi-user touch technology. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, pages 219–226, New York, NY, USA. Association for Computing Machinery.

Dipietro, L., Sabatini, A. M., and Dario, P. (2008). A Survey of Glove-Based Systems and Their Applications. *Trans. Sys. Man Cyber Part C*, 38(4):461–482.

Dominguez, S. M., Keaton, T., and Sayed, A. H. (2006). A Robust Finger Tracking Method for Multimodal Wearable Computer Interfacing. *IEEE Transactions on Multimedia*, 8(5):956–972.

Drews, P. and Weyrich, M. (1997). A system for digital mock-up's and virtual prototype design in industry: 'the Virtual Workbench'. In *ISIE '97 Proceeding of the IEEE International Symposium on Industrial Electronics*, pages 1292–1296 vol.3.

Ens, B., Byagowi, A., Han, T., Hincapié-Ramos, J. D., and Irani, P. (2016). Combining Ring Input with Hand Tracking for Precise, Natural Interaction with Spatial Analytic Interfaces. In *Proceedings of the 2016 Symposium on Spatial User Interaction*, SUI '16, pages 99–102, New York, NY, USA. ACM.

Fortmann, J., Heuten, W., and Boll, S. (2015). User Requirements for Digital Jewellery. In *Proceedings of the 2015 British HCI Conference*, British HCI '15, pages 119–125, New York, NY, USA. ACM.

Fukui, R., Watanabe, M., Gyota, T., Shimosaka, M., and Sato, T. (2011). Hand Shape Classification with a Wrist Contour Sensor: Development of a Prototype Device. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 311–314, New York, NY, USA. ACM.

Fukumoto, M. and Suenaga, Y. (1994). "FingeRing": A Full-time Wearable Interface. In *Conference Companion on Human Factors in Computing Systems*, CHI '94, pages 81–82, New York, NY, USA. ACM.

Fukumoto, M. and Tonomura, Y. (1997). &Ldquo;Body Coupled FingerRing&Rdquo;: Wireless Wearable Keyboard. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, pages 147–154, New York, NY, USA. ACM.

Gong, H., Cui, Z., Wang, Y., Shen, C., Zhang, D., and Luo, S. (2021). eGlove: Designing Interactive Fabric Sensor for Enhancing Contact-Based Interactions. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, number 275, pages 1–7. Association for Computing Machinery, New York, NY, USA.

Grosse-Puppendahl, T., Beck, S., Wilbers, D., Zeiß, S., von Wilmsdorff, J., and Kuijper, A. (2014). Ambient Gesture-Recognizing Surfaces with Visual Feedback. In Streitz, N. and Markopoulos, P., editors, *Distributed, Ambient, and Pervasive Interactions*, Lecture Notes in Computer Science, pages 97–108, Cham. Springer International Publishing.

Grosse-Puppendahl, T., Berghoefer, Y., Braun, A., Wimmer, R., and Kuijper, A. (2013a). Opencapsense: A Rapid Prototyping Toolkit for Pervasive Interaction Using Capacitive Sensing. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 152–159.

Grosse-Puppendahl, T., Braun, A., Kamieth, F., and Kuijper, A. (2013b). Swiss-cheese extended: An object recognition method for ubiquitous interfaces based on capacitive proximity sensing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1401–1410, Paris, France. Association for Computing Machinery.

Grosse-Puppendahl, T., Holz, C., Cohn, G., Wimmer, R., Bechtold, O., Hodges, S., Reynolds, M. S., and Smith, J. R. (2017). Finding Common Ground: A Survey of Capacitive Sensing in Human-Computer Interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 3293–3315, Denver, Colorado, USA. Association for Computing Machinery.

Gu, Y., Yu, C., Li, Z., Li, Z., Wei, X., and Shi, Y. (2020). QwertyRing: Text Entry on Physical Surfaces Using a Ring. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4):128:1–128:29.

Gummeson, J., Priyantha, B., and Liu, J. (2014). An Energy Harvesting Wearable Ring Platform for Gestureinput on Surfaces. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 162–175, New York, NY, USA. ACM.

Gupta, A., Ji, C., Yeo, H.-S., Quigley, A., and Vogel, D. (2019). RotoSwype: Word-Gesture Typing using a Ring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–12, Glasgow, Scotland Uk. Association for Computing Machinery.

Han, T., Hasan, K., Nakamura, K., Gomez, R., and Irani, P. (2017). SoundCraft: Enabling Spatial Interactions on Smartwatches Using Hand Generated Acoustics. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, pages 579–591, New York, NY, USA. ACM.

Haque, F., Nancel, M., and Vogel, D. (2015). Myopoint: Pointing and Clicking Using Forearm Mounted Electromyography and Inertial Motion Sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3653–3656, New York, NY, USA. ACM.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Holz, C., Grossman, T., Fitzmaurice, G., and Agur, A. (2012). Implanted User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 503–512, New York, NY, USA. ACM.

Horn, B., Hilden, H., and Negahdaripour, S. (1988). Closed-Form Solution of Absolute Orientation using Orthonormal Matrices. *Journal of the Optical Society of America A*, 5:1127–1135.

Hosono, S., Nishimura, S., Iwasaki, K., and Tamaki, E. (2019). Gesture Recognition System using Optical Muscle Deformation Sensors. In *Proceedings of the 2019 2nd International Conference on Electronics, Communications and Control Engineering*, ICECC 2019, pages 12–15, New York, NY, USA. Association for Computing Machinery.

Hrabia, C.-E., Wolf, K., and Wilhelm, M. (2013). Whole Hand Modeling Using 8 Wearable Sensors: Biomechanics for Hand Pose Prediction. In *Proceedings of the 4th Augmented Human International Conference*, AH '13, pages 21–28, New York, NY, USA. ACM.

Hsieh, Y.-T., Jylhä, A., Orso, V., Andolina, S., Hoggan, E., Gamberini, L., and Jacucci, G. (2019). Developing hand-worn input and haptic support for real-world target finding. *Personal and Ubiquitous Computing*, 23(1):117–132.

Hsieh, Y.-T., Jylhä, A., Orso, V., Gamberini, L., and Jacucci, G. (2016). Designing a Willing-to-Use-in-Public Hand Gestural Interaction Technique for Smart Glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 4203–4215, New York, NY, USA. ACM.

Hsiu, M.-C., Wang, C., Huang, D.-Y., Lin, J.-W., Lin, Y.-C., Yang, D.-N., Hung, Y.-p., and Chen, M. (2016). Nail+: Sensing Fingernail Deformation to Detect Finger Force Touch Interactions on Rigid Surfaces. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '16, pages 1–6, New York, NY, USA. ACM.

Hu, F., He, P., Xu, S., Li, Y., and Zhang, C. (2020). FingerTrak: Continuous 3D Hand Pose Tracking by Deep Learning Hand Silhouettes Captured by Miniature Thermal Cameras on Wrist. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2):71:1–71:24.

Huang, D., Zhang, X., Saponas, T. S., Fogarty, J., and Gollakota, S. (2015). Leveraging Dual-Observable Input for Fine-Grained Thumb Interaction Using Forearm EMG. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pages 523–528, New York, NY, USA. ACM.

InvenSense (2017). MPU-9250 Product Specification Revision 1.1. https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf. Accessed on 26.04.2017.

Iravantchi, Y., Goel, M., and Harrison, C. (2019). BeamBand: Hand Gesture Sensing with Ultrasonic Beamforming. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–10, Glasgow, Scotland Uk. Association for Computing Machinery.

Jain, B. and Schultz, D. (2018). Asymmetric Learning Vector Quantization for Efficient Nearest Neighbor Classification in Dynamic Time Warping Spaces. *attern Recognition*, 76:349–366.

Jing, L., Cheng, Z., Zhou, Y., Wang, J., and Huang, T. (2013). Magic Ring: A Self-contained Gesture Input Device on Finger. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, pages 39:1–39:4, New York, NY, USA. ACM.

Kao, H.-L. C., Dementyev, A., Paradiso, J. A., and Schmandt, C. (2015). NailO: Fingernails As an Input Surface. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3015–3018, New York, NY, USA. ACM.

Karasawa, H., Fukui, R., Watanabe, M., and Warisawa, S. (2019). Simultaneous Recognition of Hand Shape and Two-Axis Wrist Bending Using Wearable Wrist Contour Measuring Device. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1550–1555.

Kato, H. and Takemura, K. (2016). Hand Pose Estimation Based on Active Bone-conducted Sound Sensing. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, pages 109–112, New York, NY, USA. ACM.

Ketabdar, H., Moghadam, P., and Roshandel, M. (2012). Pingu: A New Miniature Wearable Device for Ubiquitous Computing Environments. In *2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 502–506.

Kienzle, W., Whitmire, E., Rittaler, C., and Benko, H. (2021). ElectroRing: Subtle Pinch and Touch Detection with a Ring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, number 3, pages 1–12. Association for Computing Machinery, New York, NY, USA.

Kim, D., Hilliges, O., Izadi, S., Butler, A. D., Chen, J., Oikonomidis, I., and Olivier, P. (2012). Digits: Freehand 3D Interactions Anywhere Using a Wrist-worn Gloveless Sensor. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 167–176, New York, NY, USA. ACM.

Kim, M., Jang, I., Lee, Y., Lee, Y., and Lee, D. (2016). Wearable 3-DOF cutaneous haptic device with integrated IMU-based finger tracking. In *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 649–649.

Lackerschmid, C. (2018). *Entwicklung Eines Gestenspotter Für PeriSense*. Bachelor thesis, Technische Universität Berlin, Berlin, Germany.

Laput, G. and Harrison, C. (2019). Sensing Fine-Grained Hand Activity with Smartwatches. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–13, Glasgow, Scotland Uk. Association for Computing Machinery.

Ledger, D. (2014). Inside Wearables (Part 2). Technical report, Endeavour Partners.

Lee, L. and Hui, P. (2018). Interaction Methods for Smart Glasses: A Survey. *IEEE Access*, 6:28712–28732.

Lee, S., Buxton, W., and Smith, K. C. (1985). A multi-touch three dimensional touch-sensitive tablet. *ACM SIGCHI Bulletin*, 16(4):21–25.

Lee, Y., Kim, M., Lee, Y., Kwon, J., Park, Y., and Lee, D. (2019). Wearable Finger Tracking and Cutaneous Haptic Interface with Soft Sensors for Multi-Fingered Virtual Manipulation. *IEEE/ASME Transactions on Mechatronics*, 24(1):67–77.

Liang, C., Yu, C., Qin, Y., Wang, Y., and Shi, Y. (2021). DualRing: Enabling Subtle and Expressive Hand Interaction with Dual IMU Rings. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(3):115:1–115:27.

Lim, H., Chung, J., Oh, C., Park, S., and Suh, B. (2016). OctaRing: Examining Pressure-Sensitive Multi-Touch Input on a Finger Ring Device. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16 Adjunct, pages 223–224, New York, NY, USA. ACM.

Lissermann, R., Huber, J., Hadjakos, A., and Mühlhäuser, M. (2013). EarPut: Augmenting Behind-the-ear Devices for Ear-based Interaction. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 1323–1328, New York, NY, USA. ACM.

Liu, Y., Jiang, F., and Gowda, M. (2020). Finger Gesture Tracking for Interactive Applications: A Pilot Study with Sign Languages. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):112:1–112:21.

Liu, Y., Lin, C., and Li, Z. (2021a). WR-Hand: Wearable Armband Can Track User's Hand. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(3):118:1–118:27.

Liu, Y., Zhang, S., and Gowda, M. (2021b). NeuroPose: 3D Hand Pose Tracking using EMG Wearables. In *Proceedings of the Web Conference 2021*, WWW '21, pages 1471–1482, New York, NY, USA. Association for Computing Machinery.

Lucas, J., Margo, C., Oussar, Y., and Holé, S. (2015). Physical limitations on spatial resolution in electrical capacitance tomography. *Measurement Science and Technology*, 26(12):125105.

Lyons, K. (2020). Wearable magnetic field sensing for finger tracking. In *Proceedings of the 2020 International Symposium on Wearable Computers*, ISWC '20, pages 63–67, New York, NY, USA. Association for Computing Machinery.

Ma, Y., Mao, Z.-H., Jia, W., Li, C., Yang, J., and Sun, M. (2011). Magnetic Hand Tracking for Human-Computer Interface. *IEEE Transactions on Magnetics*, 47(5):970–973.

Marnanel (2007). "Finger-spelling" alphabet. https://commons.wikimedia.org/wiki/File:Asl_alphabet_gallaudet_ann.svg.

Matthies, D. J., Woodall, A., and Urban, B. (2021). Prototyping Smart Eyewear with Capacitive Sensing for Facial and Head Gesture Detection. In *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*, UbiComp '21, pages 476–480, New York, NY, USA. Association for Computing Machinery.

McIntosh, J., Marzo, A., Fraser, M., and Phillips, C. (2017). EchoFlex: Hand Gesture Recognition Using Ultrasound Imaging. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 1923–1934, New York, NY, USA. ACM.

McIntosh, J., McNeill, C., Fraser, M., Kerber, F., Löchtefeld, M., and Krüger, A. (2016). EMPress: Practical Hand Gesture Classification with Wrist-Mounted EMG and Pressure Sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2332–2342, New York, NY, USA. ACM.

Montebaur, M., Wilhelm, M., Hessler, A., and Albayrak, S. (2020). A Gesture Control System for Drones Used with Special Operations Forces. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '20, page 77, New York, NY, USA. Association for Computing Machinery.

Murao, K. (2015). Wearable Text Input Interface Using Touch Typing Skills. In *Proceedings of the 6th Augmented Human International Conference*, AH '15, pages 207–208, New York, NY, USA. ACM.

Nandakumar, R., Iyer, V., Tan, D., and Gollakota, S. (2016). FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1515–1525, New York, NY, USA. ACM.

Nassour, J., Amirabadi, H. G., Weheabby, S., Ali, A. A., Lang, H., and Hamker, F. (2020). A Robust Data-Driven Soft Sensory Glove for Human Hand Motions Identification and Replication. *IEEE Sensors Journal*, 20(21):12972–12979.

Nguyen, V., Rupavatharam, S., Liu, L., Howard, R., and Gruteser, M. (2019). HandSense: Capacitive coupling-based dynamic, micro finger gesture recognition. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, SenSys '19, pages 285–297, New York, New York. Association for Computing Machinery.

Nirjon, S., Gummeson, J., Gelb, D., and Kim, K.-H. (2015). TypingRing: A Wearable Ring Platform for Text Input. In *Proceedings of the 13th Annual International*

Conference on Mobile Systems, Applications, and Services, MobiSys '15, pages 227–239, New York, NY, USA. ACM.

Ortega-Avila, S., Rakova, B., Sadi, S., and Mistry, P. (2015). Non-invasive Optical Detection of Hand Gestures. In *Proceedings of the 6th Augmented Human International Conference*, AH '15, pages 179–180, New York, NY, USA. ACM.

Oum, J. H., Lee, S. E., Kim, D.-W., and Hong, S. (2008). Non-contact heartbeat and respiration detector using capacitive sensor with Colpitts oscillator. *Electronics Letters*, 44(2):87–89.

Paredes, L., Reddy, S. S., Chidambaram, S., Vagholkar, D., Zhang, Y., Benes, B., and Ramani, K. (2021). FabHandWear: An End-to-End Pipeline from Design to Fabrication of Customized Functional Hand Wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(2):76:1–76:22.

Parizi, F. S., Whitmire, E., and Patel, S. (2019). AuraRing: Precise Electromagnetic Finger Tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(4):150:1–150:28.

Park, K., Kim, D., Heo, S., and Lee, G. (2020). MagTouch: Robust Finger Identification for a Smartwatch Using a Magnet Ring and a Built-in Magnetometer. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–13, New York, NY, USA. Association for Computing Machinery.

Park, K. and Lee, G. (2019). FingMag: Finger Identification Method for Smartwatch. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, pages 1–6, New York, NY, USA. Association for Computing Machinery.

Peshock, A., Duvall, J., and Dunne, L. E. (2014). Argot: A Wearable One-handed Keyboard Glove. In *Proceedings of the 2014 ACM International Symposium on Wearable Computers: Adjunct Program*, ISWC '14 Adjunct, pages 87–92, New York, NY, USA. ACM.

Pickering, C. A. (2008). Human Vehicle Interaction Based On Electric Field Sensing. In Valldorf, J. and Gessner, W., editors, *Advanced Microsystems for Automotive Applications 2008*, VDI-Buch, pages 141–154. Springer, Berlin, Heidelberg.

Platt, J. (1998). Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical report.

Poupyrev, I., Gong, N.-W., Fukuhara, S., Karagozler, M. E., Schwesig, C., and Robinson, K. E. (2016). Project Jacquard: Interactive Digital Textiles at Scale. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*,

CHI '16, pages 4216–4227, San Jose, California, USA. Association for Computing Machinery.

Rantanen, V., Venesvirta, H., Spakov, O., Verho, J., Vetek, A., Surakka, V., and Lekkala, J. (2013). Capacitive Measurement of Facial Activity Intensity. *IEEE Sensors Journal*, 13(11):4329–4338.

Rekimoto, J. (2001). Gesturewrist and Gesturepad: Unobtrusive Wearable Interaction Devices. In *Proceedings Fifth International Symposium on Wearable Computers*, pages 21–27. IEEE Comput. Soc.

Reyes, G., Wu, J., Juneja, N., Goldshtein, M., Edwards, W. K., Abowd, G. D., and Starner, T. (2018). SynchroWatch: One-Handed Synchronous Smartwatch Gestures Using Correlation and Magnetic Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4):158:1–158:26.

Röddiger, T., Beigl, M., Wolffram, D., Budde, M., and Sun, H. (2020). PDMSkin: On-Skin Gestures with Printable Ultra-Stretchable Soft Electronic Second Skin. In *Proceedings of the Augmented Humans International Conference*, AHs '20, pages 1–9, New York, NY, USA. Association for Computing Machinery.

Salemi Parizi, F., Kienzle, W., Whitmire, E., Gupta, A., and Benko, H. (2021). RotoWrist: Continuous Infrared Wrist Angle Tracking using a Wristband. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, VRST '21, pages 1–11, New York, NY, USA. Association for Computing Machinery.

Sato, M., Poupyrev, I., and Harrison, C. (2012). Touché: Enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 483–492, Austin, Texas, USA. Association for Computing Machinery.

Sharma, A., Roo, J. S., and Steimle, J. (2019). Grasping Microgestures: Eliciting Single-hand Microgestures for Handheld Objects. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–13, Glasgow, Scotland Uk. Association for Computing Machinery.

Shi, Y., Zhang, H., Zhao, K., Cao, J., Sun, M., and Nanayakkara, S. (2020). Ready, Steady, Touch! Sensing Physical Contact with a Finger-Mounted IMU. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2):59:1–59:25.

Shilkrot, R., Huber, J., Steimle, J., Nanayakkara, S., and Maes, P. (2015). Digital Digits: A Comprehensive Survey of Finger Augmentation Devices. *ACM Comput. Surv.*, 48(2):30:1–30:29.

Simmons, L. and Welsh, J. (2014). Particle Filter Based Finger Tracking Utilising Magnetoresistive Sensors. In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 560–565.

Simmons, L. P. and Welsh, J. S. (2015). Unscented Kalman Filter based finger tracking utilising magnetoresistive sensors. In *2015 American Control Conference (ACC)*, pages 5128–5133.

Singh, G., Nelson, A., Robucci, R., Patel, C., and Banerjee, N. (2015). Inviz: Low-power personalized gesture recognition using wearable textile capacitive sensor arrays. In *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 198–206.

Smith, J. R. (1996). Field Mice: Extracting Hand Geometry from Electric Field Measurements. *IBM Systems Journal*, 35(3.4):587–608.

Smith, J. R. (1999). *Electric Field Imaging*. PhD thesis, MIT Media Lab, Massachusetts.

Soliman, M., Mueller, F., Hegemann, L., Roo, J. S., Theobalt, C., and Steimle, J. (2018). FingerInput: Capturing Expressive Single-Hand Thumb-to-Finger Microgestures. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*, ISS '18, pages 177–187, New York, NY, USA. Association for Computing Machinery.

Sousa, M., Techmer, A., Steinhage, A., Lauterbach, C., and Lukowicz, P. (2013). Human tracking and identification using a sensitive floor and wearable accelerometers. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 166–171.

Spiegel, S., Jain, B.-J., and Albayrak, S. (2014). Fast Time Series Classification Under Lucky Time Warping Distance. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 71–78, New York, NY, USA. ACM.

Sridhar, S., Markussen, A., Oulasvirta, A., Theobalt, C., and Boring, S. (2017). WatchSense: On- and Above-Skin Input Sensing Through a Wearable Depth Sensor. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 3891–3902, New York, NY, USA. ACM.

Sun, W., Li, F. M., Huang, C., Lei, Z., Steeper, B., Tao, S., Tian, F., and Zhang, C. (2021). ThumbTrak: Recognizing Micro-finger Poses Using a Ring with Proximity Sensing. In *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*, MobileHCI '21, pages 1–9, New York, NY, USA. Association for Computing Machinery.

Takada, R., Kadomoto, J., and Shizuki, B. (2019). A Sensing Technique for Data Glove Using Conductive Fiber. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, pages 1–4, New York, NY, USA. Association for Computing Machinery.

Talkhestani, B. A. and Weyrich, M. (2020). Digital Twin of manufacturing systems: A case study on increasing the efficiency of reconfiguration. *at - Automatisierungstechnik*, 68(6):435–444.

Tamaki, E., Hosono, S., and Iwasaki, K. (2019). FirstVR: A Muscle Deformation Sensors Array Device to Detect Finger Gestures and Noise Reduction Case. In *Proceedings of the 2019 2nd International Conference on Electronics, Communications and Control Engineering*, ICECC 2019, pages 21–24, New York, NY, USA. Association for Computing Machinery.

TapRing (2017). Tap Wearable Keyboard - Turn Any Surface into a Keyboard. http://www.tapwithus.com/.

ten Holt, G., Reinders, M., and Hendriks, E. (2007). Multi-dimensional dynamic time warping for gesture recognition. *Annual Conference of the Advanced School for Computing and Imaging*.

TexasInstruments (2015). FDC1004 4-Channel Capacitance-to-Digital Converter for Capacitive Sensing Solutions. http://www.ti.com/lit/ds/symlink/fdc1004.pdf. Accessed on 15.06.2015.

Truong, H., Zhang, S., Muncuk, U., Nguyen, P., Bui, N., Nguyen, A., Lv, Q., Chowdhury, K., Dinh, T., and Vu, T. (2018). CapBand: Battery-free Successive Capacitance Sensing Wristband for Hand Gesture Recognition. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, pages 54–67, New York, NY, USA. Association for Computing Machinery.

Ueno, A., Akabane, Y., Kato, T., Hoshino, H., Kataoka, S., and Ishiyama, Y. (2007). Capacitive Sensing of Electrocardiographic Potential Through Cloth From the Dorsal Surface of the Body in a Supine Position: A Preliminary Study. *IEEE Transactions on Biomedical Engineering*, 54(4):759–766.

Üstün, B., Melssen, W. J., and Buydens, L. M. C. (2006). Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel. *Chemometrics and Intelligent Laboratory Systems*, 81(1):29–40.

Valtonen, M., Kivimäki, T., and Vanhala, J. (2012). Capacitive 3D user tracking with a mobile demonstration platform. In *Proceeding of the 16th International Academic MindTrek Conference*, MindTrek '12, pages 61–63, New York, NY, USA. Association for Computing Machinery.

Voelker, S., Cherek, C., Thar, J., Karrer, T., Thoresen, C., Øvergård, K. I., and Borchers, J. (2015). PERCs: Persistently Trackable Tangibles on Capacitive Multi-Touch Displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pages 351–356, New York, NY, USA. Association for Computing Machinery.

Wang, C.-Y., Chu, W.-C., Chiu, P.-T., Hsiu, M.-C., Chiang, Y.-H., and Chen, M. Y. (2015a). PalmType: Using Palms As Keyboards for Smart Glasses. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 153–160, New York, NY, USA. ACM.

Wang, C.-Y., Hsiu, M.-C., Chiu, P.-T., Chang, C.-H., Chan, L., Chen, B.-Y., and Chen, M. Y. (2015b). PalmGesture: Using Palms As Gesture Interfaces for Eyes-free Input. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 217–226, New York, NY, USA. ACM.

Wang, D. (2015). Capacitive Sensing: Ins and Outs of Active Shielding. http://www.ti.com/lit/an/snoa926a/snoa926a.pdf. Accessed on 06.05.2015.

Watanabe, H. and Terada, T. (2018). Improving ultrasound-based gesture recognition using a partially shielded single microphone. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, ISWC '18, pages 9–16, Singapore, Singapore. Association for Computing Machinery.

Weiser, M. (1999). The Computer for the 21st Century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11.

Wilhelm, M., Krakowczyk, D., Trollmann, F., and Albayrak, S. (2015). eRing: Multiple Finger Gesture Recognition with One Ring Using an Electric Field. In *Proceedings of the 2Nd International Workshop on Sensor-based Activity Recognition and Interaction*, WOAR '15, pages 7:1–7:6, New York, NY, USA. ACM.

Wimmer, R. (2011). Capacitive Sensors for Whole Body Interaction. In England, D., editor, *Whole Body Interaction*, pages 121–133. Springer London, London.

Wimmer, R., Kranz, M., Boring, S., and Schmidt, A. (2007a). A Capacitive Sensing Toolkit for Pervasive Activity Detection and Recognition. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'07)*, pages 171–180.

Wimmer, R., Kranz, M., Boring, S., and Schmidt, A. (2007b). CapTable and CapShelf - Unobtrusive Activity Recognition Using Networked Capacitive Sensors. In *2007 Fourth International Conference on Networked Sensing Systems*, pages 85–88.

Withana, A., Groeger, D., and Steimle, J. (2018). Tacttoo: A Thin and Feel-Through Tattoo for On-Skin Tactile Output. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 365–378, Berlin, Germany. Association for Computing Machinery.

Wobbrock, J. O., Wilson, A. D., and Li, Y. (2007). Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 159–168, Newport, Rhode Island, USA. Association for Computing Machinery.

Wolf, K., Naumann, A., Rohs, M., and Müller, J. (2011). Taxonomy of Microinteractions: Defining Microgestures Based on Ergonomic and Scenario-dependent Requirements. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part I*, INTERACT'11, pages 559–575, Berlin, Heidelberg. Springer-Verlag.

Wolf, K., Schleicher, R., Kratz, S., and Rohs, M. (2013). Tickle: A Surface-independent Interaction Technique for Grasp Interfaces. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*, TEI '13, pages 185–192, New York, NY, USA. ACM.

Wu, E., Yuan, Y., Yeo, H.-S., Quigley, A., Koike, H., and Kitani, K. M. (2020a). Back-Hand-Pose: 3D Hand Pose Estimation for a Wrist-worn Camera via Dorsum Deformation Network. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, pages 1147–1160, New York, NY, USA. Association for Computing Machinery.

Wu, J., Colglazier, C., Ravishankar, A., Duan, Y., Wang, Y., Ploetz, T., and Starner, T. (2018a). Seesaw: Rapid one-handed synchronous gesture interface for smartwatches. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, ISWC '18, pages 17–20, New York, NY, USA. Association for Computing Machinery.

Wu, J.-F., Qiu, C., Wang, Y., Zhao, R., Cai, Z.-P., Zhao, X.-G., He, S.-S., Wang, F., Wang, Q., and Li, J.-Q. (2018b). Human Limb Motion Detection with Novel Flexible Capacitive Angle Sensor Based on Conductive Textile. *Electronics*, 7(9):192.

Wu, T., Fukuhara, S., Gillian, N., Sundara-Rajan, K., and Poupyrev, I. (2020b). ZebraSense: A Double-sided Textile Touch Sensor for Smart Clothing. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, pages 662–674, New York, NY, USA. Association for Computing Machinery.

Xi, X., Keogh, E., Shelton, C., Wei, L., and Ratanamahatana, C. A. (2006). Fast Time Series Classification Using Numerosity Reduction. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 1033–1040, New York, NY, USA. ACM.

Yamato, Y., Suzuki, Y., Sekimori, K., Shizuki, B., and Takahashi, S. (2020). Hand Gesture Interaction with a Low-Resolution Infrared Image Sensor on an Inner Wrist. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '20, pages 1–5, New York, NY, USA. Association for Computing Machinery.

Yeo, H.-S., Lee, J., Bianchi, A., Samboy, A., Koike, H., Woo, W., and Quigley, A. (2020). WristLens: Enabling Single-Handed Surface Gesture Interaction for Wrist-Worn Devices Using Optical Motion Sensor. In *Proceedings of the Augmented Humans International Conference*, AHs '20, pages 1–8, New York, NY, USA. Association for Computing Machinery.

Yeo, H.-S., Lee, J., Kim, H.-i., Gupta, A., Bianchi, A., Vogel, D., Koike, H., Woo, W., and Quigley, A. (2019a). WRIST: Watch-Ring Interaction and Sensing Technique for Wrist Gestures and Macro-Micro Pointing. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '19, pages 1–15, New York, NY, USA. Association for Computing Machinery.

Yeo, H.-S., Wu, E., Lee, J., Quigley, A., and Koike, H. (2019b). Opisthenar: Hand Poses and Finger Tapping Recognition by Observing Back of Hand Using Embedded Wrist Camera. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, pages 963–971, New Orleans, LA, USA. Association for Computing Machinery.

Yokota, T. and Hashida, T. (2016). Hand Gesture and On-body Touch Recognition by Active Acoustic Sensing Throughout the Human Body. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16 Adjunct, pages 113–115, New York, NY, USA. ACM.

Yoon, S. H., Huo, K., Nguyen, V. P., and Ramani, K. (2015). TIMMi: Finger-worn Textile Input Device with Multimodal Sensing in Mobile Interaction. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '15, pages 269–272, New York, NY, USA. ACM.

Younas, J., Margarito, H., Bian, S., and Lukowicz, P. (2020). Finger Air Writing - Movement Reconstruction with Low-cost IMU Sensor. In *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '20, pages 69–75, New York, NY, USA. Association for Computing Machinery.

Zhang, B., Chen, Y., Qian, Y., and Wang, X. (2011). A Ring-shaped Interactive Device for Large Remote Display and Mobile Device Control. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 473–474, New York, NY, USA. ACM.

Zhang, C., Waghmare, A., Kundra, P., Pu, Y., Gilliland, S., Ploetz, T., Starner, T. E., Inan, O. T., and Abowd, G. D. (2017a). FingerSound: Recognizing Unistroke Thumb Gestures Using a Ring. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3):120:1–120:19.

Zhang, C., Wang, X., Waghmare, A., Jain, S., Ploetz, T., Inan, O. T., Starner, T. E., and Abowd, G. D. (2017b). FingOrbits: Interaction with Wearables Using Synchronized Thumb Movements. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ISWC '17, pages 62–65, New York, NY, USA. ACM.

Zhang, T., Zeng, X., Zhang, Y., Sun, K., Wang, Y., and Chen, Y. (2020). ThermalRing: Gesture and Tag Inputs Enabled by a Thermal Imaging Smart Ring. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–13, New York, NY, USA. Association for Computing Machinery.

Zhang, Y., Chen, Y., Yu, H., Yang, X., Lu, W., and Liu, H. (2018). Wearing-independent hand gesture recognition method based on EMG armband. *Personal and Ubiquitous Computing*, 22(3):511–524.

Zhang, Y. and Harrison, C. (2015). Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pages 167–173, New York, NY, USA. ACM.

Zhang, Y., Laput, G., and Harrison, C. (2017c). Electrick: Low-Cost Touch Sensing Using Electric Field Tomography. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 1–14, New York, NY, USA. Association for Computing Machinery.

Zhang, Y., Zhou, J., Laput, G., and Harrison, C. (2016). SkinTrack: Using the Body As an Electrical Waveguide for Continuous Finger Tracking on the Skin. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1491–1503, New York, NY, USA. ACM.

Zhou, J., Zhang, Y., Laput, G., and Harrison, C. (2016). AuraSense: Enabling Expressive Around-Smartwatch Interactions with Electric Field Sensing. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 81–86, New York, NY, USA. ACM.

Zimmerman, T. G., Smith, J. R., Paradiso, J. A., Allport, D., and Gershenfeld, N. (1995). Applying electric field sensing to human-computer interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 280–287, USA. ACM Press/Addison-Wesley Publishing Co.

Appendices

A. Prototype History

The attractiveness of rings as an interaction device has long been given. In the joint project Universal Home Control Interfaces@Connected Usability (UHCI) funded by the German Federal Ministry of Economics (BMWi), a printed ring was equipped with an accelerometer in 2013 (Figure A.1). The sensor was connected via a cable to an Arduino Nano[1]. This ring was designed and implemented to control smart home devices.

[1] Arduino Nano website: https://store.arduino.cc/arduino-nano (accessed on 01.09.2020)

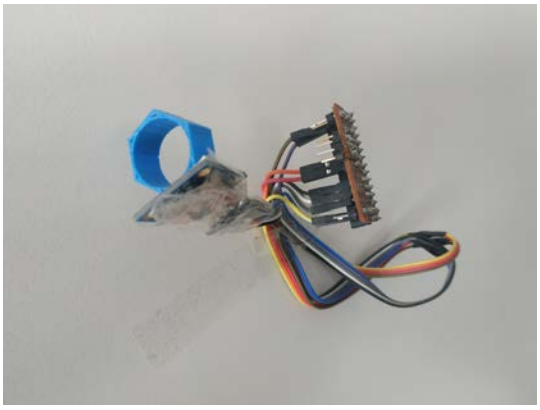


Figure A.1.: How everything started. First ring prototype using an accelerometer for gesture recognition.

Figure A.2.: Nod prototypes.

Figure A.3.: Left, a strong magnet symbolizing experiments with magnetic fields. Right, a circuit to transmit and measure radio waves.

In the later course of the project, three prototypes of the Kickstarter project Nod[2] (Figure A.2) were bought, which contained an motion sensor, and buttons.

Interactions with these rings were limited and usually limited to the detection of hand movements. The idea was born to expand the interaction space by recognizing the movements of several fingers using a field. First experiments were performed with magnetic fields (Figure A.3 left). However, it soon became apparent that this would require the use of a magnetometer on each finger and magnets on the wrist (see also (Simmons and Welsh, 2014)). Afterwards experiments with radio waves were performed (Figure A.3 right). It became obvious that the technology is difficult to implement in a ring with the given capabilities and means.

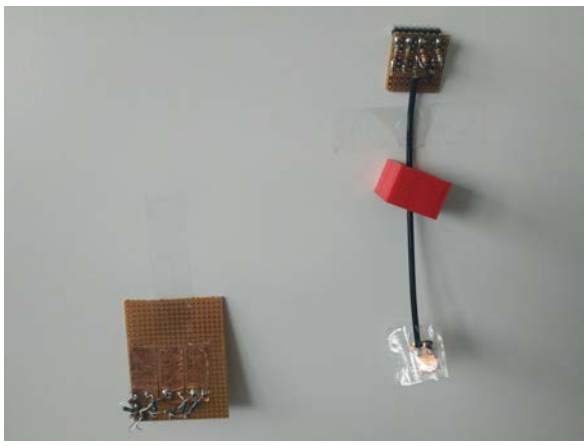
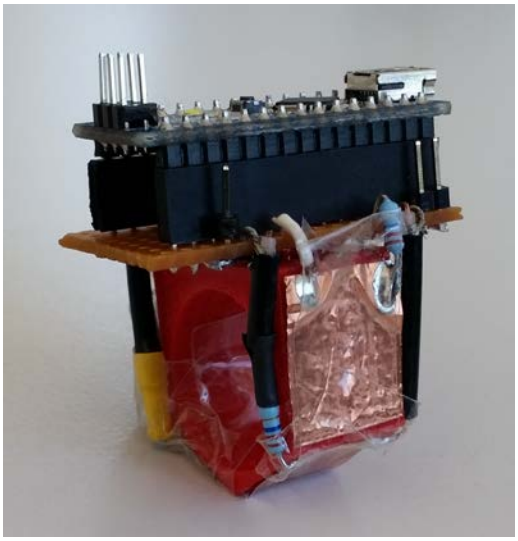


Figure A.4.: Left, three touch electrodes. Right, one electrode on a 3D-printed ring.

[2]Blog entry about the Nod ring: https://www.punchkick.com/blog/2014/04/29/nod-smart-gesture-control-ring (accessed on 01.09.2020). The ring did not reach market maturity.



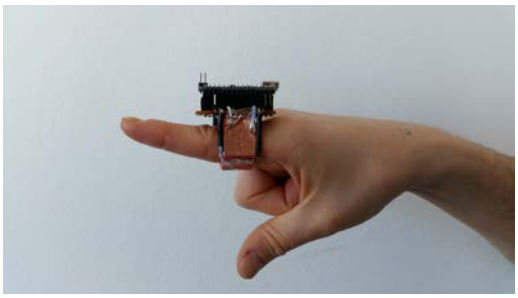


Figure A.5.: eRing prototype.

Next, first experiments with capacitive sensing were performed. It started with touch electrodes (Figure A.4 left) and later with electrodes on a rectangular ring (Figure A.4 right). The electrodes were each connected to an Arduino Nano. The electrodes were charged via the nano. It was measured how much time it takes to saturate the electrode. If this was not possible within two seconds, the measurement and charging was aborted and started again or with the next electrode.

On this basis, the first real ring prototype called eRing (Figure A.5) was developed (Section 3.4).

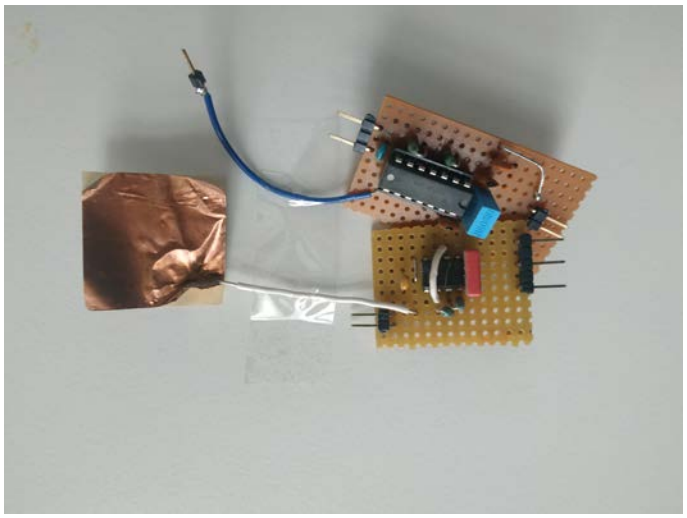


Figure A.6.: First experiments with alternating fields.

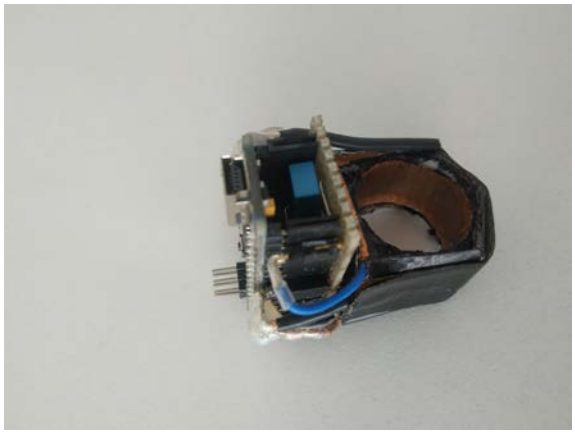


Figure A.7.: First prototype using alternating fields.

However, the measurement is very slow, the fields are too small, and the noise is too high. Furthermore, the circuit is too susceptible, especially the solder joints. It has already been shown that an integrated solution is necessary. Furthermore, the ring is not self-sufficient. The USB cable for power supply and data transfer causes interference and induces parasitic capacities.

To speed up the measurement, increase the field range, and reduce noise, the measuring principle has been changed. The electrode is now charged with a high-frequency alternating voltage (instead of direct voltage), and thus an alternating field is generated. Conductive objects within the range of the field change the frequency of the field. This frequency change can be determined. Figure A.6 shows the first experiments with this measurement technique. In the next step, this circuit was integrated into a ring (Figure A.7). It was shown that the circuit needs much space. There was only enough space to operate one electrode.

Figure A.8.: Initial version of PeriSense.

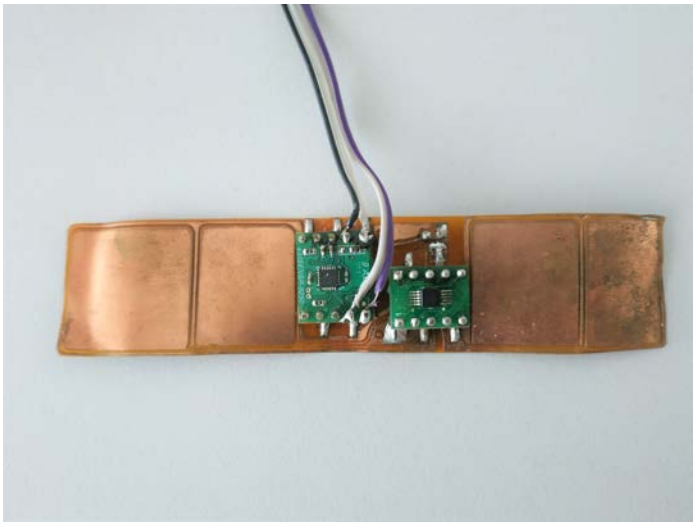


Figure A.9.: Etched electrode layout.

In order to reduce the ring size and to be able to integrate all components and operate all electrodes, a board layout was designed (Figure A.8). The board already comes with all connectors to be extended with a Bluetooth module, additional sensors, and a battery. The resulting prototype is the initial version of PeriSense.

It was found that the cables are constantly touched by finger and hand. Despite shielded cables, interference occurs. Also, the soldered connections of the cables are much too susceptible. Therefore, in the next version the electrodes and the traces for electrodes were etched (Figure A.9 and A.10).

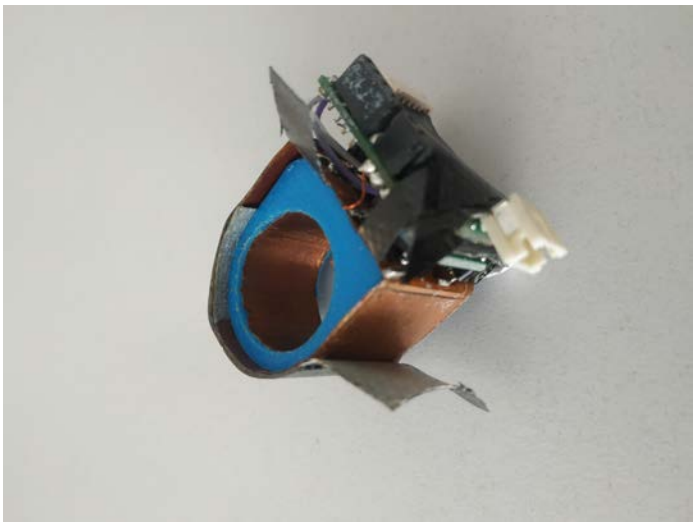


Figure A.10.: Prototype using an etched electrode layout.

Figure A.11.: First self-sufficient PeriSense prototype with TI FDC1004, Invensense MPU9250, Bluetooth module, and Battery.

However, the measurement was still very noisy and inaccurate. To simplify and speed up the development, the decision was made to use the Texas Instrument FDC1004 sensor (TexasInstruments, 2015) for the capacitive measurement. Also, an Invensense MPU9250 motion sensor (InvenSense, 2017) and a Bluetooth module were also integrated. This version was also equipped with a rechargeable battery. Thus, this version was the first self-sufficient PeriSense version (Figure A.11), which could be tested with the first users.

The solder connections and connectors cause loose contacts and leakage currents, strongly influencing the capacitive measurements. Therefore a flexible board layout was designed and manufactured. All sensors are now integrated on this board. This PeriSense version is the one presented and evaluated in this thesis (Figure A.12).

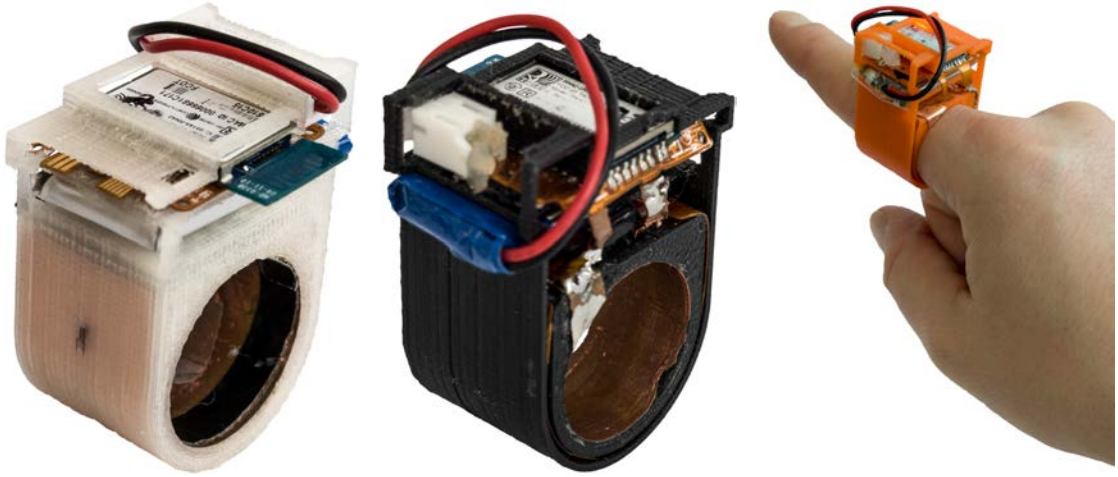


Figure A.12.: Final PeriSense prototypes in different colors and sizes.

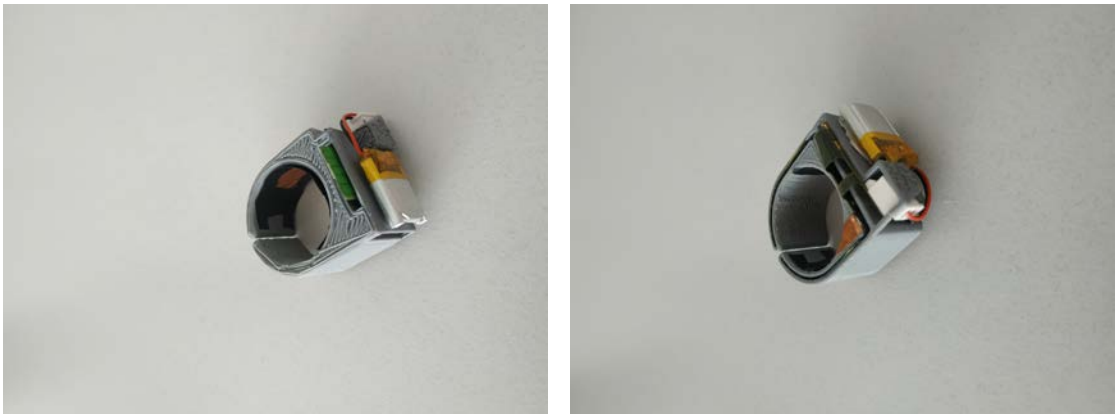


Figure A.13.: The third ring generation.

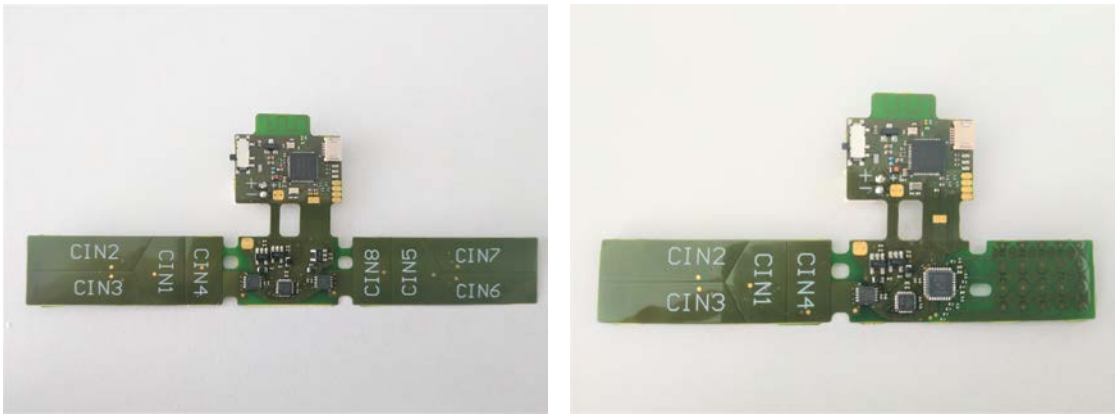


Figure A.14.: Left, the new flexible board layout and right a new board layout with a touch matrix.

Based on the experiences of this dissertation, a new generation of rings was developed (Figure A.13). They are equipped with an ARM Cortex M4 microcontroller with an integrated Bluetooth low energy module[3], two TI FDC10004 sensors, and eight electrodes in a 2-dimensional arrangement. Furthermore, the ring has become smaller. Another version was also designed, which has a touch matrix on the thumb side (Figure A.14). An evaluation of these prototypes is still pending.

All evolutionary stages were tested and developed beforehand with pegboards, as shown as an example in Figure A.15. Figure A.16 shows the evolution of ring shapes over time, and figure A.17 shows all ring prototype versions above my former desktop at the DAI Labor again.

[3]Nordic nRF52832: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52832 (accessed on 01.09.2020)

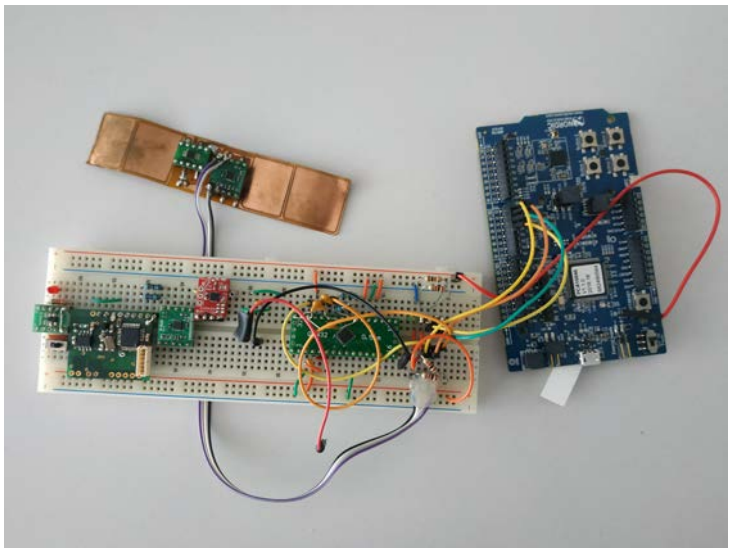


Figure A.15.: Experimental test setup for the circuits.

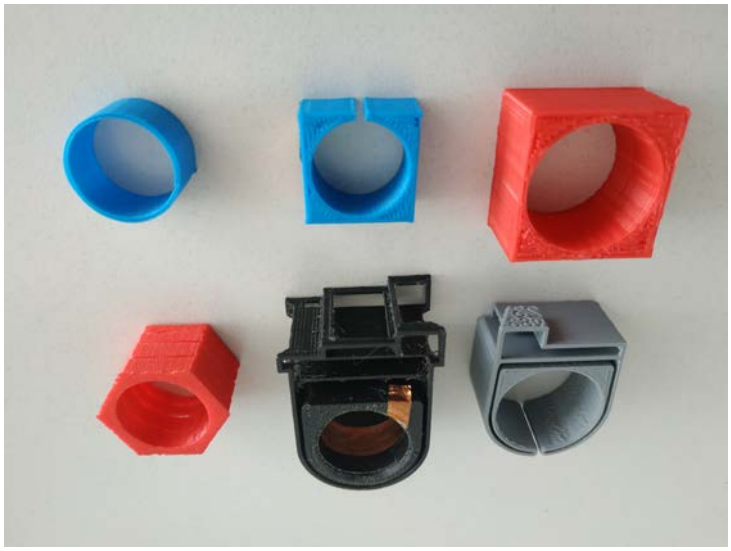


Figure A.16.: Evolution of ring shapes over time.

Figure A.17.: Prototype version gallery.

B. Layouts

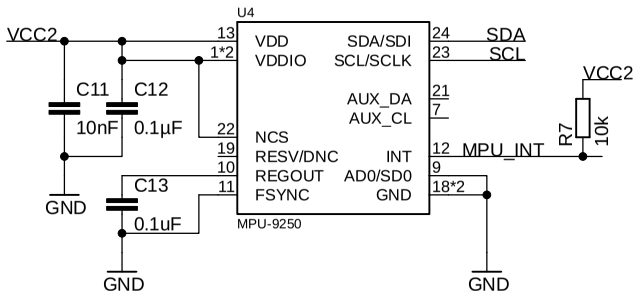
B.1. Circuit

Table B.1.: Components

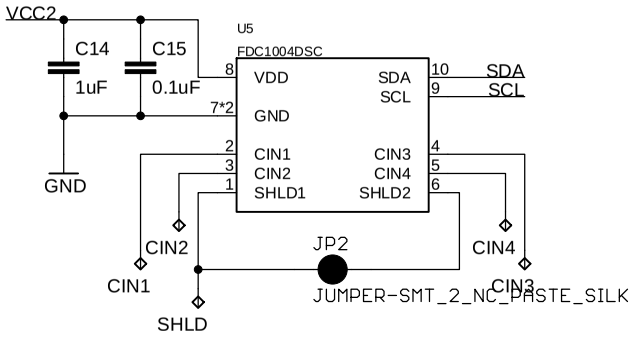
Part ID	**Value**	**Device**	**Package**
C1	0.1uF	C0201	C0201
C2	0.1uF	C0201	C0201
C3	0.1uF	C0201	C0201
C4	0.1uF	C0201	C0201
C5	4.7μF	CPOL0603	C0603
C6	0.1μF	C0201	C0201
C7	22pF	C0201	C0201
C8	22pF	C0201	C0201
C9	0.1μF	C0201	C0201
C10	1uF	C0201	C0201
C11	10nF	C0201	C0201
C12	0.1μF	C0201	C0201
C13	0.1uF	C0201	C0201
C14	1uF	C0201	C0201
C15	0.1uF	C0201	C0201
C20	1uF	C0201	C0201
C21	1uF	C0201	C0201
C22	1uF	C0201	C0201
C23	10μF	CPOL0603	C0603
C24	0.1μF	C0201	C0201
CIN1	TPTP06R NOSTOP	TPTP06R NOSTOP	TP06R NOSTOP
CIN2	TPTP06R NOSTOP	TPTP06R NOSTOP	TP06R NOSTOP
CIN3	TPTP06R NOSTOP	TPTP06R NOSTOP	TP06R NOSTOP
CIN4	TPTP06R NOSTOP	TPTP06R NOSTOP	TP06R NOSTOP
CON1		CON06	CON06
JP2	JUMPER-SMT 2 NC PASTE SILK	JUMPER-SMT 2 NC PASTE SILK	SMT-JUMPER 2 NC PASTE SILK

JP RX	JUMPER-SMT 3 1-NC PASTE SILK	JUMPER-SMT 3 1-NC PASTE SILK	SMT-JUMPER 3 1-NC PASTE SILK
JP TX	JUMPER-SMT 3 1-NC PASTE SILK	JUMPER-SMT 3 1-NC PASTE SILK	SMT-JUMPER 3 1-NC PASTE SILK
LED B		LEDSML0603	SML0603
LED G		LEDSML0603	SML0603
LED R		LEDSML0603	SML0603
P1	BATT	FFC-03P1.0MM	FFC-03P-1.0MM
P2	ISP	FFC-06P0.5MM	FFC-06P-0.5MM
P3	I2C	FFC-04P1.0MM	FFC-04P-1.0MM
P4	UART DBG	FFC-03P1.0MM	FFC-03P-1.0MM
P5	GND	FFC-03P1.0MM	FFC-03P-1.0MM
R1	1k	R0201	R0201
R2	10k	R0201	R0201
R3	10k	R0201	R0201
R4	470	R0201	R0201
R5	470	R0201	R0201
R6	470	R0201	R0201
R7	10k	R0201	R0201
R8	1k	R0201	R0201
S1	SWITCH-SPDT-GND	SWITCH-SPDT-GND	SWITCH SPST SMD
SHLD	TPTP06R NOSTOP	TPTP06R NOSTOP	TP06R NOSTOP
U$ 1	FFC-CONN-03P1.0MM	FFC-CONN-03P1.0MM	FFC-CONN-03P-1.0MM
U$ 2	FFC-CONN-04P1.0MM	FFC-CONN-04P1.0MM	FFC-CONN-04P-1.0MM
U$ 3	FFC-CONN-06P0.5MM	FFC-CONN-06P0.5MM	FFC-CONN-06P-0.5MM
U$ 4	C-GRID-04	C-GRID-04	SL-LP-4-SMD-005
U$ 5	C-GRID-03	C-GRID-03	SL-LP-3-SMD-005
U1	MCP1700T-3002 E/TT	MCP1703CB	SOT23
U2	MCP1700T-3002 E/TT	MCP1703CB	SOT23
U3	ATMEGA328P VQFNVQFN	ATMEGA328P VQFNVQFN	VQFN-28
U4	MPU-9250	MPU-9250	QFN.24.4X4.NO EP
U5	FDC1004DSC	FDC1004DSC	DSC.10
U7	BLUETOOTH-RN-41	BLUETOOTH-RN-41	RN41
X1		XTALNX5032	NX5032

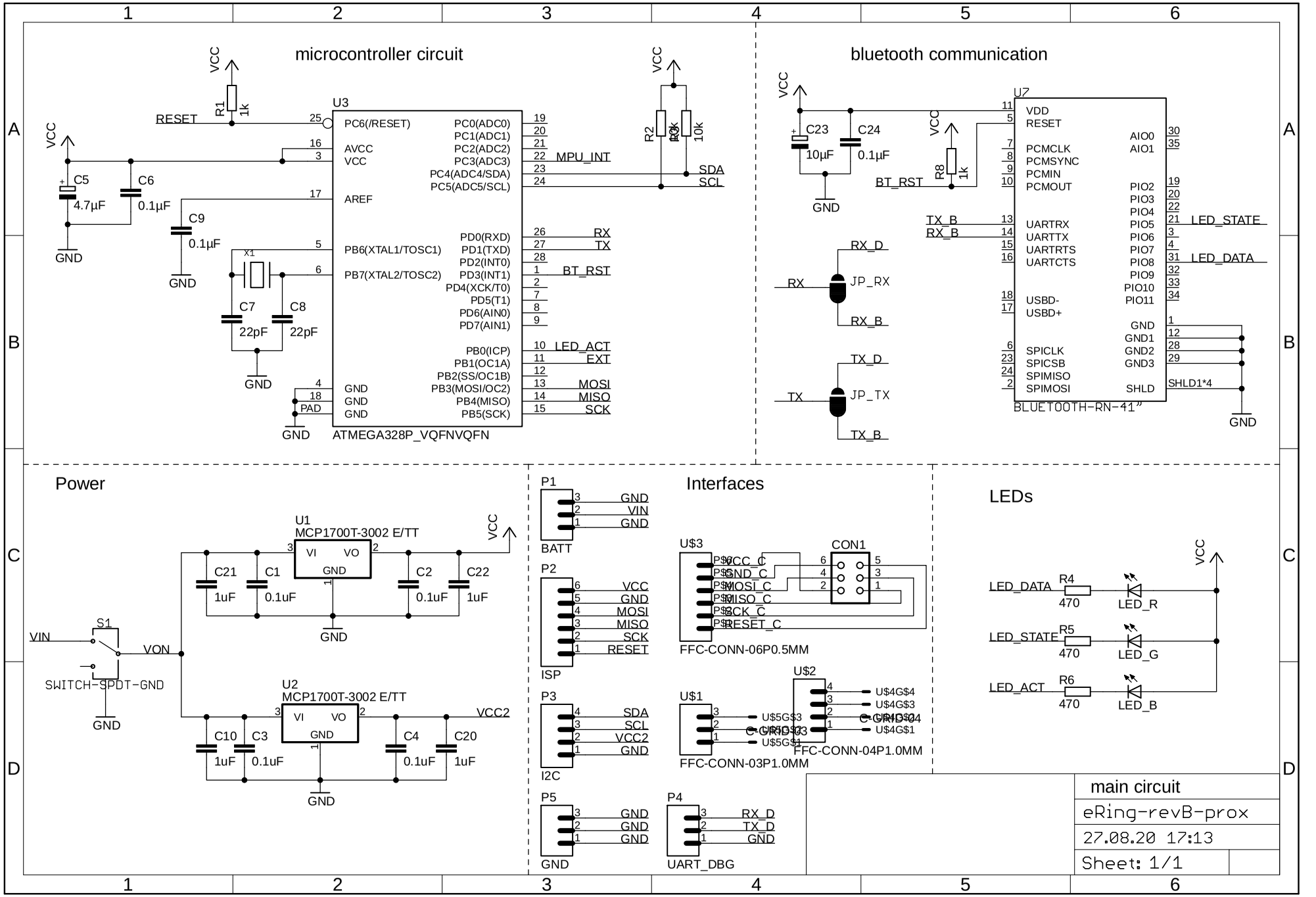
Inertial Measurement Unit



Capacitive Proximity Sensing



	eRing-revB-prox
	27.08.20 17:13
	Sheet: 1/1


microcontroller circuit
bluetooth communication
Power
Interfaces
LEDs
U3
ATMEGA328P_VQFNVQFN
U7
BLUETOOTH-RN-41"
U1
MCP1700T-3002 E/TT
U2
MCP1700T-3002 E/TT
SWITCH-SPDT-GND
FFC-CONN-06P0.5MM
FFC-CONN-03P1.0MM
FFC-CONN-04P1.0MM
BATT
ISP
I2C
UART_DBG
GND
main circuit
eRing-revB-prox
27.08.20 17:13
Sheet: 1/1

B.2. Layouts

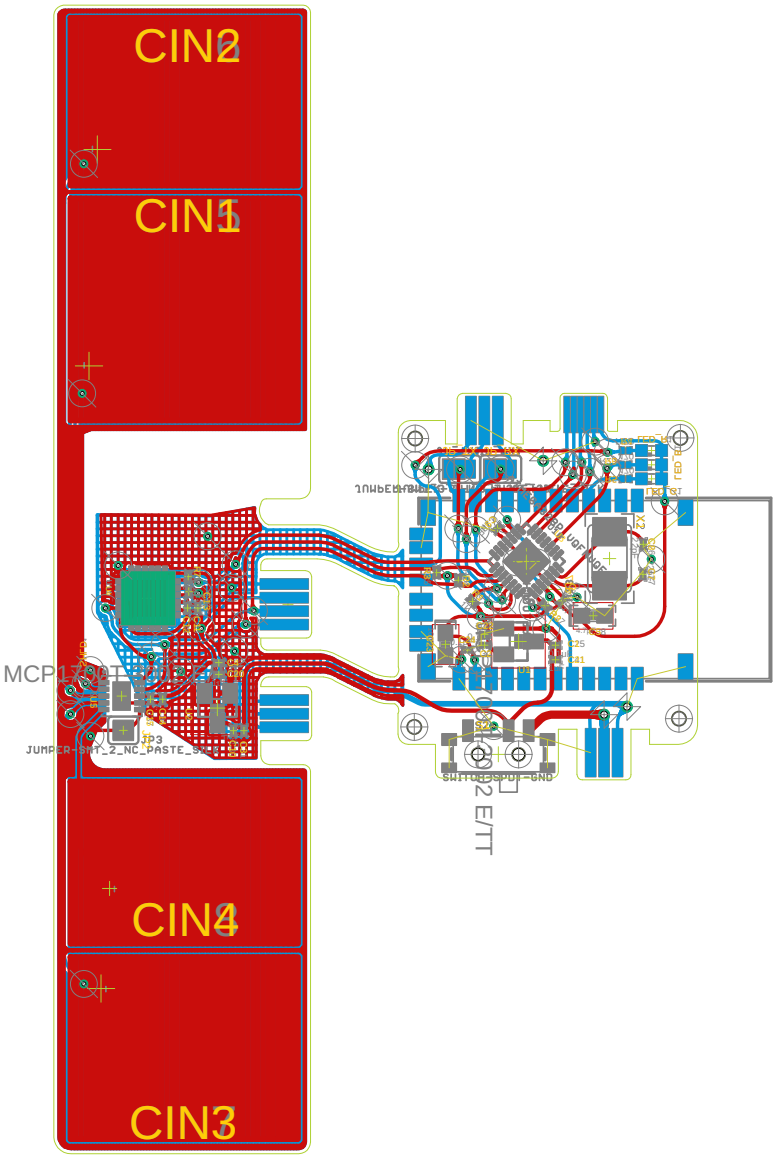


Figure B.1.: Board layout with all layers.

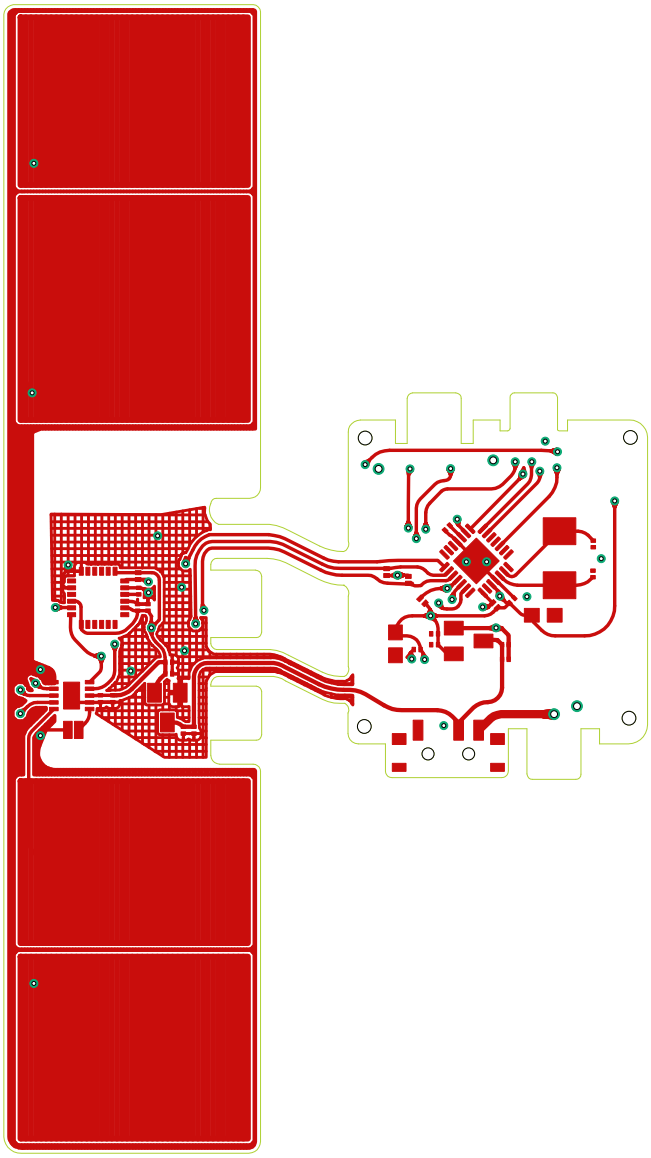


Figure B.2.: Top layer.

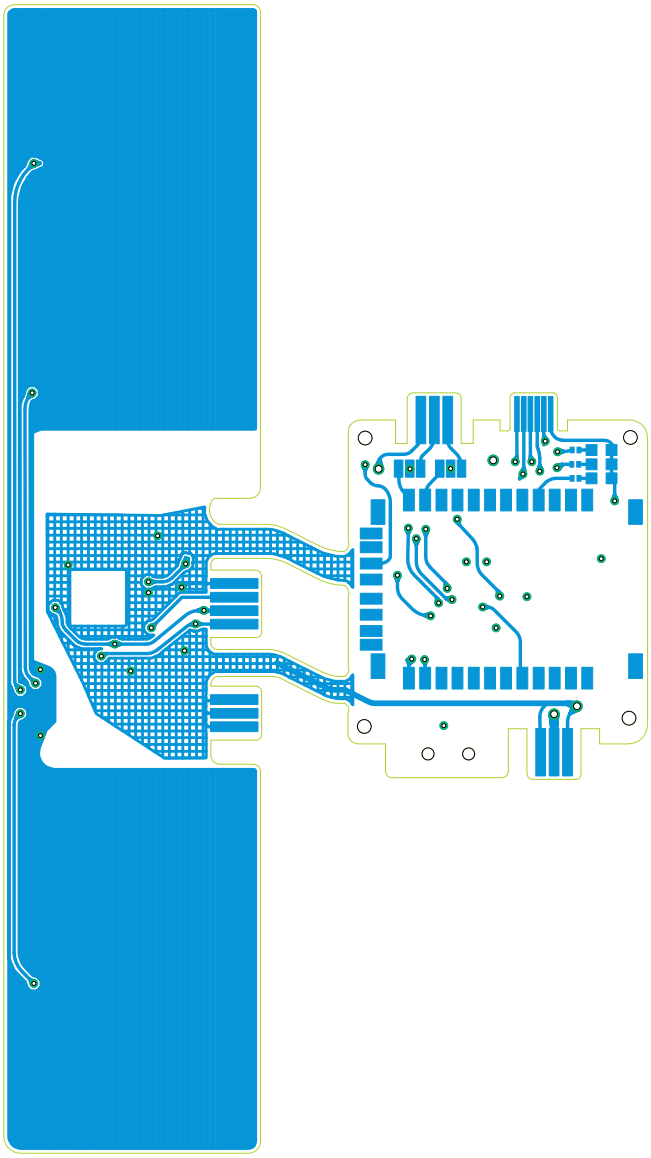


Figure B.3.: Bottom layer.

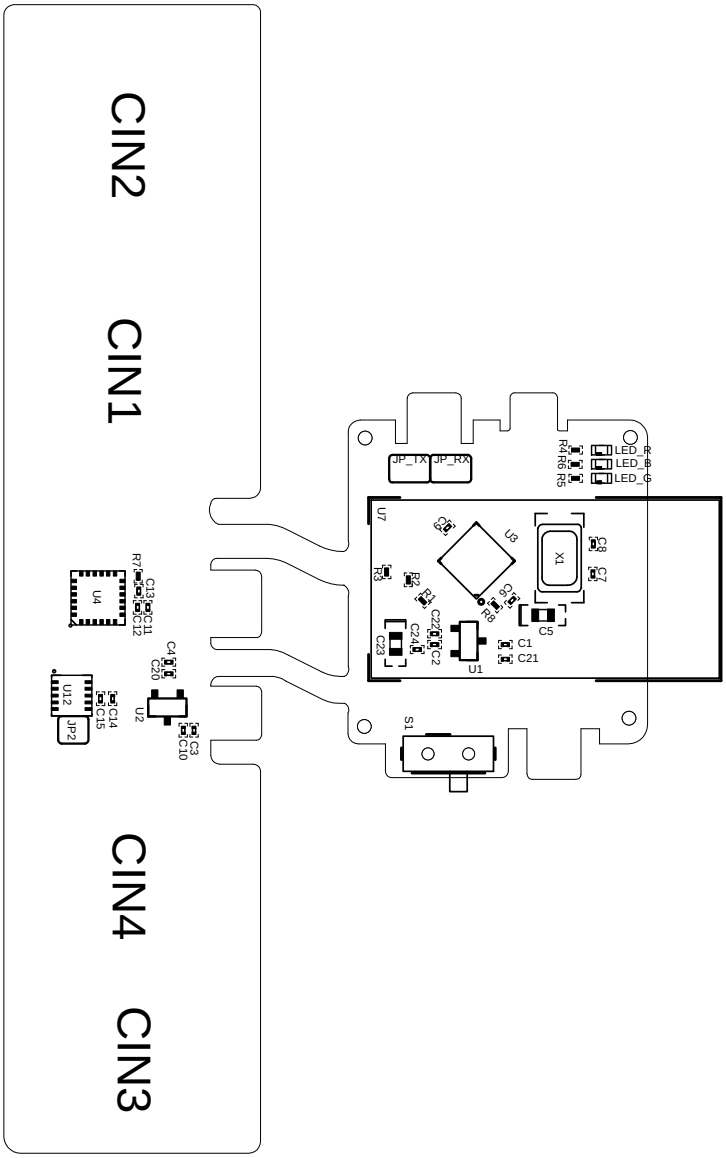


Figure B.4.: Layer with documented components.

C. PeriSensePy Example Configuration

This appendix presents a sample implementation of an interaction pipeline using the PeriSensePy framework. The interaction pipeline is described in the following (Section C.1), and then the corresponding JSON configuration (Section C.2) and how it is executed are presented (Section C.3).

C.1. Interaction Pipeline

Figure C.1 shows an example of an interaction pipeline. The interaction detection is based on a sliding window of size 10, which is shifted by 5 steps. A feature vector is computed for each window, which serves as input for a Support Vector Classification (SVC). If a window is classified as a gesture, the sensor data is buffered by the gesture recognition. As soon as a window is classified as noise, the buffering is stopped, and the buffered sequence as a whole is classified by gesture recognition. The gesture recognition is based on a 1-nearest neighbor classifier (1NN) that uses Dynamic Time Warping (DTW) as a similarity measure. Using a generalized learning vector quantization (GLVQ), the training set size is reduced by computing three prototypes per class. The detected gesture is sent to the IOLITE platform via an MQTT channel. In addition, the detected gesture and the interaction detection events are printed to the console.

C.2. Configuration

In the following, an example configuration file of the interaction pipeline described above is presented.

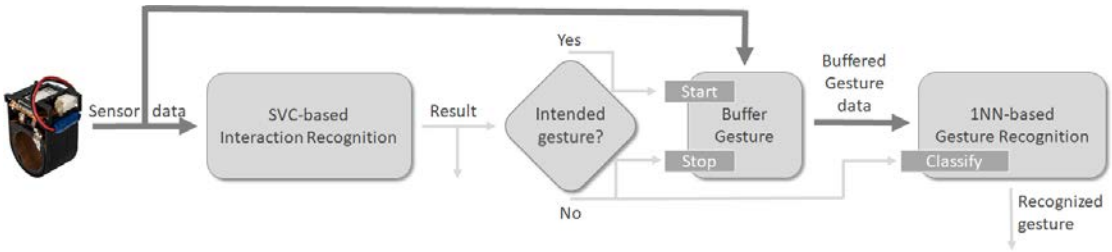


Figure C.1.: Example interaction pipeline.

```
{
"device" : {
  "driver" : "PeriSense",
  "whiteList" : [ ],
  "blackList" : [ ],
  "scan_timeout" : 30
},
"InteractionModeRecognition" : {
  "filter2" : {
    "comment" : "",
    "id" : "filter2",
    "class_name" : "FeatureBasedFilter",
    "active" : true,
    "grIds" : [ "gesrec2" ],
    "features" : {
      "timestamp" : false,
      "b" : false,
      "e1" : false,
      "e2" : false,
      "e3" : false,
      "e4" : false,
      "e5" : false,
      "e6" : false,
      "e7" : false,
      "e8" : false,
      "ax" : true,
      "ay" : true,
      "az" : true,
      "gx" : true,
      "gy" : true,
      "gz" : true,
      "temp" : false },
    "buffer_size" : 15,
    "window_length" : 10,
    "step_width" : 5,
    "fs" : 50,
    "model_file" : "svm_acc_gestures_w10_step2.model",
    "feature_conf_file" : "fcfg.json",
    "training" : {
      "train" : true,
      "classifier" : 'SVC',
```

```
42      "gesture_train_files" : [ "gestures_10_01.csv", "gestures_10_02.
           csv" ],
43      "noise_train_files" : [ "noise_10_01.csv", "noise_10_02.csv" ] }
           ,
44    }
45  },
46 "GestureRecognition" : {
47   "recognizer2" : {
48     "comment" : "knn",
49     "id" : "gesrec2",
50     "class_name" : "OneNN",
51     "active" : true,
52     "features" : {
53       "timestamp" : false,
54       "b" : false,
55       "e1" : false,
56       "e2" : false,
57       "e3" : false,
58       "e4" : false,
59       "e5" : false,
60       "e6" : false,
61       "e7" : false,
62       "e8" : false,
63       "ax" : true,
64       "ay" : true,
65       "az" : true,
66       "gx" : true,
67       "gy" : true,
68       "gz" : true,
69       "temp" : false },
70     "gestures" : [
71       { "label" : 1,
72       "name" : "up" },
73       { "label" : 2,
74       "name" : "down" },
75       { "label" : 3,
76       "name" : "left" },
77       { "label" : 4,
78       "name" : "right" },
79       { "label" : 5,
80       "name" : "action } ],
```

```
    "buffer_size" : 15,
    "window_length" : 0,
    "step_width" : 1,
    "n_to_add" : 15,
    "n_to_delete" : 5,
    "max_window_length" : 35,
    "min_window_length" : 8,
    "norm_file" : "norm.model",
    "model_file" : "oneNN_dglvq.model",
    "rec_threshold" : 1.0,
    "recognition_break" : 1.0,
    "training" : {
      "train" : true,
      "measure" : "DTW",
      "normalization" : "norm",
      "min_values" : [ -16, -16, -16, -2000, -2000, -2000 ],
      "max_values" : [ 16, 16, 16,2000,  2000, 2000 ],
      "optimizing" : {
        "optimizer" : "DGLVQ",
        "k_prototypes" : 3,
        "A_mean_algo" : 1,
        "learning_rate" : 20.01 },
      "train_files" : [ "gestures_10_01.csv", "gestures_10_02.csv" ] }
  }
},
"OutputModules" : {
  "PrintConsole1" : {
    "id" : "debugOutput",
    "class_name" : "PrintConsole",
    "recIds" : [ "gesrec2", "filter2" ],
    "mapping_only" : false,
    "gesture_mapping" : { },
    "active" : false
  },
  "iolite" : {
    "id" : "ioliteOutput",
    "class_name" : "MQTT_IOLITE",
    "address" : "127.0.0.1",
    "port" : 1883,
    "recIds" : [ "gesrec2" ],
    "mapping_only" : false,
```

```
122     "run_forever" : false,
123     "gesture_mapping" : {
124       "action" : "SELECT",
125       "left" : "LEFT",
126       "right" : "RIGHT",
127       "up" : "UP",
128       "down" : "DOWN" },
129     "active" : true
130   }
131 }
132 }
```

C.3. Run Interaction Pipeline

The above presented pipeline is executed as following:

```
$ sudo python3 run_demo.py -c config.hjson
```

or alternatively in the python console:

```
> from demos.ApplicationPipeline import ApplicationPipeline
> ap = ApplicationPipeline("config.hjson")
> ap.start()
```

On Linux-based operating systems, it is necessary to run this command with root privileges since the PeriSense driver's Bluetooth access requires this.

To train the models defined in the configuration, the following command is to execute:

```
$ python3 run_train.py -c config.hjson
```

or alternatively in the python console:

```
> from demos.TrainingPipeline import TrainingPipeline
> tr = TrainingPipeline("config.hjson")
> tr.train()
```