Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Grablets: Enabling Twist and Fold Interactions with Textile Icons

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Julian Wallerius

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Dr.-Ing. Heiko Müller

Registration date: 12.06.2024
Submission date: 14.10.2024

Zentrales Prüfungsamt/Central Examination Office

RWTHAACHEN UNIVERSITY

Eidesstattliche Versicherung
Declaration of Academic Integrity

Name, Vorname/Last Name, First Name	Matrikelnummer (freiwillige Angabe) Student ID Number (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare under penalty of perjury that I have completed the present paper/bachelor's thesis/master's thesis* entitled

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt; dies umfasst insbesondere auch Software und Dienste zur Sprach-, Text- und Medienproduktion. Ich erkläre, dass für den Fall, dass die Arbeit in unterschiedlichen Formen eingereicht wird (z.B. elektronisch, gedruckt, geplottet, auf einem Datenträger) alle eingereichten Versionen vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without unauthorized assistance from third parties (in particular academic ghostwriting**.** I have not used any other sources or aids than those indicated; this includes in particular software and services for language, text, and media production. In the event that the work is submitted in different formats (e.g. electronically, printed, plotted, on a data carrier), I declare that all the submitted versions are fully identical. I have not previously submitted this work, either in the same or a similar form to an examination body.

Ort, Datum/City, Date	Unterschrift/Signature

*Nichtzutreffendes bitte streichen/Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
§ 156 StGB (German Criminal Code): False Unsworn Declarations
Whosoever before a public authority competent to administer unsworn declarations (including Declarations of Academic Integrity) falsely submits such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment for a term not exceeding three years or to a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
§ 161 StGB (German Criminal Code): False Unsworn Declarations Due to Negligence
(1) If an individual commits one of the offenses listed in §§ 154 to 156 due to negligence, they are liable to imprisonment for a term not exceeding one year or to a fine.
(2) The offender shall be exempt from liability if they correct their false testimony in time. The provisions of § 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Ort, Datum/City, Date	Unterschrift/Signature

Contents

List of Figures and Tables

Abstract

Textile interfaces are a promising possibility for integrating controls of digital devices into the environment. Textiles have the advantage of being deformable and stretchable, which allows for more complex interactions than just touching. We present Grablets, graspable icons embedded into textiles that allow folding and twisting input. We integrated an Inertial Measurement Unit (IMU) into an icon and encompassed the icon into a stretchable fabric. Utilizing a neural network we trained, we then developed a data pipeline to detect which gestures were performed. Our user study showed that users generally like the Grablet but are confused about the exact interaction with our current prototype. The study provided many ideas for improving the interaction's affordance and comfort. We also present some ideas about why the performance of our machine-learning model was insufficient for actual use and how to enhance it in the future.

Überblick

Textile Interfaces stellen eine vielversprechende Möglichkeit dar, Steuerungen von digitalen Geräten in die Umgebung zu integrieren. Textilien haben den Vorteil, verformbar und dehnbar zu sein, was komplexere Interaktionen erlaubt als bloßes Berühren. Wir präsentieren Grablets, greifbare Icons, die in Stoff eingebettet sind und Falt- und Dreheingaben erlauben. Wir integrierten eine inertiale Messeinheit in ein Icon und fassten dieses Icon in dehnbaren Stoff ein. Mithilfe eines von uns trainierten neuronalen Netzwerkes entwickelten wir dann eine Datenpipeline, um zu erkennen, welche Gesten ausgeführt wurden. Unsere Studie zeigte, dass Nutzer generell das Grablet mögen, es aber Verwirrung über die genaue Interaktion mit unserem aktuellen Prototyp gibt. Die Studie hat viele Ideen hervorgebracht, wie die Interaktion komfortabler gestaltet und die Möglichkeit der Interaktion besser hervorgehoben werden kann. Wir präsentieren außerdem einige Ideen, warum die Leistung des auf maschinellem Lernen beruhenden Modells nicht für den realen Gebrauch ausreichend war und wie diese in Zukunft verbessert werden kann.

Acknowledgments

Firstly, I would like to thank Prof. Dr. Jan Borchers and Dr.-Ing. Heiko Müller for examining my thesis.

Special thanks to my supervisor, Oliver Nowak, for the constant advice, support, and guidance throughout the whole process.

Additionally, I want to thank everyone from the chair who made this a pleasant experience. I am particularly thankful to Robert, Esra, Lennart, and Lars for supporting me when technical difficulties arose.

I also want to thank all my study participants for volunteering their time to participate in the study and providing me with crucial feedback.

Lastly, I want to express my gratitude to my friends and family for their support and patience with me during this time.

Conventions

Throughout this thesis, we use the following conventions:

- The thesis is written in American English.
- The first person is written in plural form.
- For unidentified third persons, we use they/their.

Short excursuses are set off in colored boxes.

EXCURSUS:
Excursuses are set off in orange boxes.

Where appropriate, paragraphs are summarized by one or two sentences that are positioned at the margin of the page.

This is a summary of a paragraph.

Statistical measures like the standard deviation and accuracy are rounded to two decimal places.

Chapter 1

Introduction

Since technology surrounds us everywhere, the search for new interaction types and ways to control it becomes increasingly important. Having the controls on the device itself raises a reachability problem since the user must get to the device. While remote controllers try to solve this problem, they can also be out of reach or misplaced. Instead, integrating controls into everyday objects and the environment, making them disappear from the foreground of the surroundings, is a better solution. Substituting a dedicated controlling device with such a controller removes an avoidable device and solves the problem of misplacement. Voice assistants, for example, allow interaction from everywhere within a certain radius but also introduce new hurdles. Since there is no physical user interface, there is no knowledge in the world, following the division by Norman [2013]. Instead, all knowledge is in the user's head, requiring them to remember which commands exist and what they do. Furthermore, the necessity of speaking out loud to control something can be inappropriate or disruptive, even in a smart home environment, for instance, when watching a quiet movie.

We aim to integrate controls for devices into the environment.

The idea remains of integrating the controller into the environment without significantly changing its appearance or, in other words, enhancing a surface by allowing interaction with it to control the surrounding technology. Because of their ubiquity, textiles have a lot of potential when consid-

Textiles are a promising surface candidate for interfaces.

ering different surface candidates and have been a promising part of the field of Human-Computer Interaction. The application cases for textile interfaces range from furniture (chairs, couches, curtains, and table runners) to wearables (clothes, accessories, and bags) and most other textile surface. Another advantage of textiles from most other surfaces is that they allow more ways of interaction with the user than just touching. Depending on the chosen fabric, textiles can be grasped and deformed, leading to many different ways of interaction. For example, Parzer et al. [2017] developed a sleeve that can detect multiple touch and swipe gestures and the textile being twisted, folded, bent, grasped, twirled, stretched, pushed, or shaken.

The deformable characteristic of textiles is an opportunity for new ways of interaction.

To solve the problem of voice assistants requiring the user to know all commands, the interface should give cues to the user. Particularly when a surface that has had nothing to do with controlling technology becomes interactive, something needs to signal the user how to interact with it. Icons are well-suited for signifying the possibility of interaction since they are language-independent, take up little space, and create an easy-to-understand user interface. For example, a plus and a minus sign can be used to adjust the volume of music or the brightness of lighting. Instead of only having a visual component, making the icon perceptible and distinguishable from others by touch enables eyes-free interaction. Additionally, users can explore the interface and search for the right element, eyes-free, without accidentally triggering the touch sensor.

Icons are good signifiers for textile interfaces.

The haptic component of textile icons enables eyes-free interaction.

In this thesis, we present Grablets, graspable icons embedded into textiles that allow folding and twisting input. With twisting, we refer to rotating the icon around the axis orthogonal to the plane of the textile, either clockwise or anticlockwise. Accordingly, with folding we mean rotating the icon around the two axes parallel to the textile, thus creating a fold in the fabric. We chose to pick the four directions right, left, front, and back for the folding gesture. All gestures have in common that they start with grasping the icon with at least two fingers. We describe how we integrated a sensor into a textile icon and what the data pipeline looks like. For gesture detection, we will present how we trained a neural network model to determine which gesture the

We present Grablets, graspable textile icons for fold and twist interactions.

user performed. Finally, we will evaluate our work with a small user study.

In the following of this thesis, we will discuss the related work regarding textile interfaces in Chapter 2. In Chapter 3, we present the sensor we chose, the printed circuit board we designed, and how we fabricated the textile icon with the sensor inside. The data pipeline is divided into two chapters: The first one, Chapter 4, focuses on how we communicate with the sensor, obtain data from it, and how the data is preprocessed before classification. Subsequently, we present the neural network we developed and trained and how we use it for continuous gesture classification in Chapter 5. We evaluated our prototype by conducting a user study, the procedure and results of which are the focus of Chapter 6. Finally, we talk about the limitations of our prototype and the study and what future work can be done to build upon our findings in Chapter 7. In the end, Chapter 8 concludes our thesis.

Chapter 2

Related Work

Many research approaches aim to add interactability to textiles. First, we want to examine the design aspect of textile interfaces to enable optimal user interaction. Mlakar and Haller [2020] developed some general design recommendations for textile interfaces, such as using height to differentiate between elements. Nowak et al. [2022] explored the design and fabrication of textile sliders. In addition to the placement of tick marks, they focused primarily on different shapes and height profiles of sliders to determine the best way to support the sliding gesture. Similarly, Schäfer et al. [2023] investigated the design of textile icons by producing fourteen different shapes in six different fabrication variants (varying in height and affected area) and measuring how well users could haptically recognize these types. Besides discovering confusion patterns for textile icons, they concluded design guidelines, such as using raised icons if possible, and described the fabrication process. Challis and Edwards [2001] developed guidelines for tactile interfaces also applicable to textile interfaces, like avoiding an excess of empty space and that tactile objects should be simple.

Some research aims at defining design guidelines for textile interfaces.

Many research projects focus on detecting touch and similar gestures on textile interfaces. Rekimoto [2001] built the early GesturePad prototype and integrated capacitive sensing into clothing. In recent years, Heller et al. [2014] developed pads for the upper thigh that the user can draw sim-

ple gestures on. The prototype layers a pressure-sensitive, resistive fabric between two sheets of conductive fabric. When pressure is applied, the outer layers get closer to each other, resulting in a measurable change in current. This is a pretty common type of sensor for touch detection in textiles. Xu et al. [2022] use a similar system for human activity recognition instead of a user interface. Having stripes of conductive fabric stacked on each other results in a matrix of pressure measurements. With an extensive data-processing pipeline and a machine-learning classifier, the prototype can differentiate between 18 activities.
Another system utilizing this type of textile sensor for something different than a user interface is the SensorSleeve by Randell et al. [2005] that can detect affectionate gestures like stroking the arm. More projects with this sensor are the GestureSleeve by Schneegass and Voit [2016], where the user can draw simple gestures on their lower arm to control their smartwatch, expanding its input surface, or FlexTiles by Parzer et al. [2016], a stretchable textile cover that can be used both for wearables and for furniture. ZebraSense is a slightly different, dual-sided touch sensor by Wu et al. [2020] inside a cuff that can detect the touch input on both sides of the fabric. Further projects to mention regarding conductive yarn are Post and Orth [1997] with early developments for clothing, in more recent years Poupyrev et al. [2016] with Project Jacquard, and Aigner et al. [2020] who further investigated embroidering pressure sensitive sensors. ClothTiles by Muthukumarana et al. [2021] takes a different approach. It is a prototyping framework that uses 3D printing on fabric to create simple shape-shifting interfaces.

The smart home offers many applications for textile interfaces.

There are also many textile interface approaches designed for smart homes. Brauner et al. [2017] developed an adjustable recliner armchair with a textile interface in the armrest to control its state. Another application is a cushion, developed by Suzuki et al. [2020], that can detect different gestures the user performs with it. Heller et al. [2016] built a functioning curtain prototype that automatically opens when touching it in certain spots.

Also, some approaches utilize the deformable properties of the textiles. This is particularly interesting for us since the

Grablet is meant to be moved around in the fabric. Probably the most extensive set of different interactions is the SmartSleeve system by Parzer et al. [2017]. Apart from twelve different surface gestures, which usually consist of touching and swiping, their smart sleeve can also detect the user twisting, pushing, bending, twirling, stretching, folding, grasping, and shaking the textile. It uses a pressure-sensitive textile sensor integrated into the fabric similar to Heller et al. [2014]. The matrix of pressure readings resulting from this sensor gets further processed and is finally classified into one of the gestures above with a support vector machine. Running the gesture detection continuously allows a real-time estimation of which gestures are performed.

The deformability of textiles can be used for many interactions.

Karrer et al. [2011] built a textile user interface element for clothing called Pinstripe. It utilizes the deformable nature of the textile to detect pinching the fabric and rolling it between two fingers. It is designed to be operated one-handed and is hardly activated by accident because it does not detect touch at all. The interaction includes grasping the fabric, creating a fold, and moving it around. This change in the size of the fold generates a continuous output that, for example, can be used to controll the volume of music. The prototype was realized by sewing parallel lines of conductive thread into the sleeve of a T-shirt. These threads would contact each other when the fabric was pinched.

Hamdan et al. [2016] present another approach that uses fold-based interaction with textiles. It measures at which angle the user creates a fold and is meant to be worn as a sleeve on the forearm. It was discovered, however, that users can only reliably grab at a set angle if the step between two angles is at least 30° to 45°. In addition to that study, they also built a prototype to detect grabbing similar to the sensor used for Pinstripe. Instead of parallel lines, hexagonal pads of conductive thread were integrated into the textile and distributed evenly over a small area to detect folds in any direction.

Lastly, Gioberto et al. [2013] developed a system that allows the detection of fabric bends only with a stitched sensor using a conductive thread. When the stitch is bent or folded far enough to contact itself, again the change in resistance can be measured. In testing with an animatronic mannequin and a human, the sensor performed well in terms of detection consistency.

Chapter 3

Fabrication

The choice of sensor type is crucial since it significantly influences the data processing pipeline and the way of fabricating the icon. As presented in Chapter 2, most approaches to adding interactability to textiles utilize a touch or pressure sensor. To use the strategy of making the detection of pressure on the whole fabric possible, like Parzer et al. [2017] did for SmartSleeve, would require creating the whole fabric around the icon instead of only adding the icon itself.

We chose a 9-DOF IMU to detect the orientation of the Grablet.

For this project, we want to detect the orientation and movement of the icon, so we use a nine-degrees-of-freedom inertial measurement unit (9-DOF IMU). It combines three sensors: an accelerometer to measure acceleration, a gyroscope to measure rotational motion, and a magnetometer to measure the strength of the Earth's magnetic field. In contrast to SmartSleeve, we now measure the icon's orientation rather than the fabric's deformation. However, the downside is that we must integrate an electrical component into the fabric instead of relying on conductive yarn. We will explain how we use the data from these sensors to detect the interactions in Chapter 4 and Chapter 5. Here, we focus on the process of fabricating such a textile icon, which can be divided into two steps: We first manufactured a custom printed circuit board (PCB) with the sensor and then integrated it into a textile icon.

3.1 The Circuit Board

The development board with the sensor is too big to be integrated into a textile icon.

We selected the ICM-20948 from Invensense[1] as our sensor of choice. It is an affordable, small, and low-power sensor that is also available on a development board by Adafruit[2], which allowed us to prototype easily and quickly. The development board has a size of 25.7 mm $\times$ 17.7 mm $\times$ 4.6 mm, while the sensor itself is only 3 mm $\times$ 3 mm $\times$ 1 mm big. Mlakar and Haller [2020] concluded that the optimal shape size for textile interfaces is 13 mm or bigger. In their recognition experiments, they used a size of 18 mm, just like Schäfer et al. [2023] did in their extensive study of different fabrication variants. It becomes clear that integrating the development board into a reasonable-sized textile icon would not work.

We use the Adafruit development board as a basis for our PCB design.

We, therefore, wanted to redesign Adafruit's board by eliminating unused features to decrease its size. Adafruit provides their PCB files on GitHub[3] for the Autodesk Eagle software[4]. For modification, we imported them into the KiCAD program[5], version 8.0.4. Since the sensor is also available as a surface-mount device (SMD), we can put it onto a custom PCB ourselves.

We removed the unnecessary pins and features of the Adafruit board.

The first things we removed from the Adafruit board were eight of the twelve pins and the connectors that allow it to be connected to other Adafruit boards. We will communicate with the sensor via the I^2C bus, which only requires two wires: serial clock (SCL) and serial data (SDA). Together with ground (GND) and supply voltage (VCC) for power, we arrive at a minimum of four wires needed for communication. This led to removing many electrical components regarding the features connected to these pins, such as the support for communicating with the sensor via the SPI bus.

[1] `https://invensense.tdk.com/products/motion-tracking/9-axis/icm-20948/` as of 09.2024
[2] `https://www.adafruit.com/product/4554` as of 09.2024
[3] `https://github.com/adafruit/Adafruit-ICM20948-PCB` as of 09.2024
[4] `http://eagle.autodesk.com/` as of 09.2024
[5] `https://www.kicad.org/` as of 09.2024

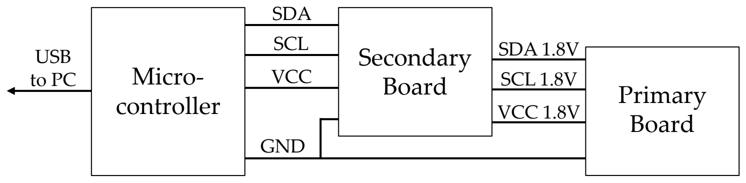


Figure 3.1: The wiring between the microcontroller, the secondary PCB that handles the voltage conversion, and the primary PCB with the sensor.

The main aspects our board retains from the original are guaranteeing a steady power supply with capacitors and converting the voltage. The ICM-20948 chip runs on 1.71V to 3.6V, while its digital in- and output voltage is in the range of 1.71V to 1.95V. Since most microcontroller boards do not use a voltage of about 1.8V for communication, and only some can provide a supply voltage in the required range, we kept the voltage converter and the system that shifts the SCL and SDA signals to 1.8V for the sensor. By that, the microcontroller can provide the board with supply voltage and communicate via the I^2C with it, but can use whatever reasonable operating and output voltage it has. The most common are 3.3V or 5V.

Our PCB keeps the circuits for a steady power supply and voltage conversion.

We decided to split the board into two because we wanted to make the PCB in the icon as small as possible: The primary board holds the sensor and only a few mandatory electrical components to integrate into the icon. The secondary board has all the voltage conversion circuits. The VCC, SDA, and SCL lines from the microcontroller are then connected to the secondary board, and the converted results are connected to the primary board. How the boards and the microcontroller connect can also be seen in Figure 3.1. Apart from making the board that needs to be integrated into the icon smaller, fabrication is also easier since both boards only require one layer. This enabled us to make the boards ourselves using a PCB mill and SMD soldering. A comparison between the original board by Adafruit and our boards can be seen in Figure 3.2.

We split the PCB into two to make the main one smaller.

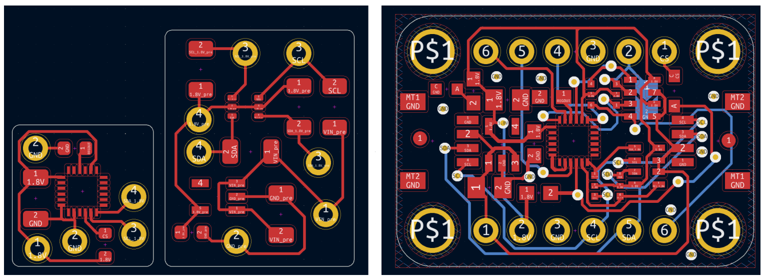


Figure 3.2: Comparison between our PCBs (left) and the original Adafruit PCB (right). It can be seen that many components, as well as the backside (blue), could be removed and the board size minimized.

The primary board has a greatly reduced size compared to the board of Adafruit.

The primary board now comes to a size of 10 mm × 10 mm × 3 mm. With this size, it can fit into some textile icons Mlakar and Haller [2020] and Schäfer et al. [2023] described. Of course, that depends on the shape of the icon. While icons like a square, circle, house, or plus might work, icons like a minus, telephone, or question mark are probably not feasible. It would be possible to change the board's shape to an elongated rectangle to some extent to integrate it into narrower icons. A professional production of the PCB may also lead to a reduced size. The secondary board has a size of 13 mm × 17 mm × 3 mm. We did not prioritize minimizing its size in the design process since it will not be integrated into the icon. The CAD files can be found in our Git repository[6].

We can still use the Adafruit libraries with our modified board.

Also, since our modification of the original Adafruit board does not change anything about the communication with the sensor itself, all of the software presented with this work also functions with their board. Conversely, we can also use all the libraries that Adafruit created for their board. How we receive data from the sensor will be the focus of Chapter 4.

[6] `https://git.rwth-aachen.de/i10/thesis/thesis-julian-wallerius-grablets`

3.2 Integration into Textile Icons

We chose a raised textile icon for the Grablet.

Now that we have a small PCB with the sensor, we need to integrate it into a textile icon. For a type of textile icon, we chose what Schäfer et al. [2023] called a *Raised Filled* fabrication variant: The whole icon is raised from the base surface. Since they found that this type of icon is the easiest to recognize eyes-free, matching with the findings of Mlakar and Haller [2020] that height is the easiest contrast to recognize on textile interfaces, and we need a bit of space to put our PCB into, this was an easy choice. We will now explain all the different parts and layers of the prototype, as shown in Figure 3.3.

We 3D-printed a case for the PCB in the shape of a plus.

We 3D-printed a case in the shape of an icon with an inlet as deep as possible for the sensor. The model for it can be found in our Git repository[7]. This cavity dictates a minimum size for the case and again excludes narrow shapes. The case is meant to give the icon its shape and protect the PCB with the sensor from excessive user pressure. We chose a plus icon because it seemed the most intuitive way to signify the affordance of folding it in four directions. Because of its many corners, it is also easy to grab and twist, unlike a circle. Our case has the size of 2.4 cm × 2.4 cm × 0.7 cm. While 7 mm seems a bit high, especially compared to the 1.6 mm Schäfer et al. [2023] used for their icons of this type, we found that a more considerable height leads to a better grip on the sides of the icon for all gestures. It also allows us to comfortably fixate the PCB and the wires from it with hot glue to the inside of the case. Additionally, the hot glue helps to stabilize the wires on the bottom of the PCB so they are less prone to break when moving the icon.

We fixated the PCB with hot glue inside the case.

The icon was inserted into a pocket of stretchable fabric.

Since the fabric needs to be stretchable to allow the fold and twist interactions we want to enable, we chose a fabric with high flexibility.[8] It consists of 78% polyester and 22% elastane and has a weight of 250 g/m^2. We layered two

[7] `https://git.rwth-aachen.de/i10/thesis/thesis-julian-wallerius-grablets`

[8] `https://www.stoffe-hemmers.de/sportjersey-altrosa` as of 09.2024

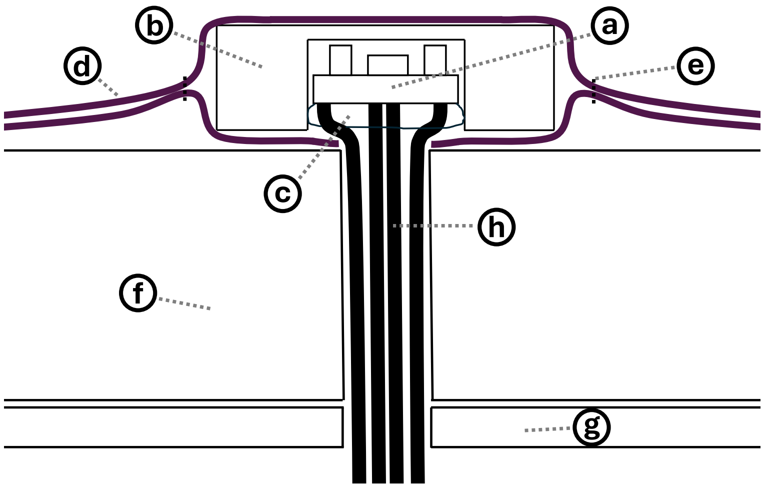


Figure 3.3: A true-to-scale cross-section of the final prototype: The custom PCB with the IMU on it (a) is fixed inside the 3D-printed icon shape (b) with hot glue (c). That is enclosed in two layers of stretchable fabric (d), the bottom of which has a small hole in the center, sewn together (e) in the form of the icon. All of that is placed on top of a sheet of upholstery foam (f) and a wooden board (g), both with a hole in the middle for the wires (h) to escape to the bottom, from where they are connected to the secondary board or the microcontroller.

pieces of this fabric and sewed them together in the form of the 3D-printed icon using an automated sewing machine. This effectively created a little sealed pocket in the fabric in the shape of the icon. We did, however, put a 2 mm of offset on all sides of the stitch to give the fabric some space to wrap around the icon. To insert the icon into this pocket, we made a small hole in the bottom sheet in the middle of the pocket. Since the fabric is very stretchable, the hole did not have to be very big to fit the whole icon through, so the icon is also improbable to fall out by accident. After that, the hole in the fabric can be used to insert and glue the PCB into its case and let the wires escape through it.

The icon and the fabric were layered on top of upholstery foam.

The fabric with the embedded icon was then put onto a 15 cm × 15 cm big, 2 cm thick sheet of upholstery foam to create a soft surface resembling, for example, an armrest

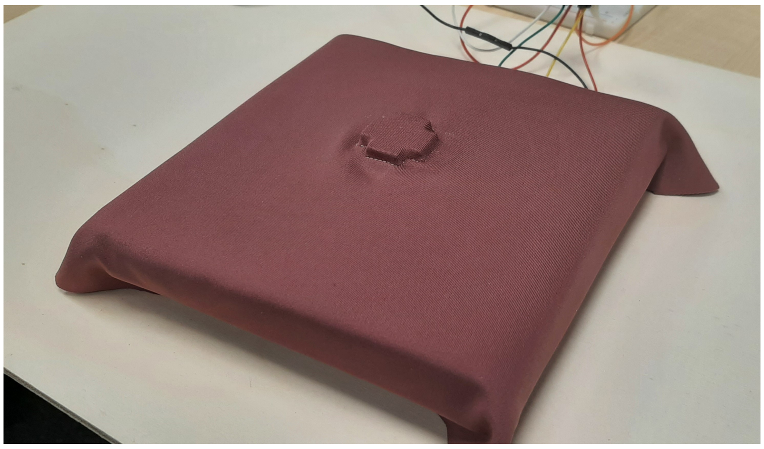


Figure 3.4: The final Grablet prototype

of a couch. It is also beneficial to be able to press one side of the icon a bit into the surface for the folding interaction. All of this was put on top of a wooden board of the same size to have a solid base plate. The foam and the base plate have a small hole through which all wires are led to the bottom, from where they are connected to the secondary PCB and the microcontroller. A square of this size, with a considerable amount of fabric on all sides of the icon, guarantees that the icon can be moved with relative ease. We folded the fabric tightly around the edges of the foam and the board and tacked it onto the backside of the wooden board. We were careful not to stretch the fabric too much in this process to keep as much potential for stretching for the gestures. Figure 3.4 shows the final prototype.

We fixated the fabric on the edges.

Chapter 4

Obtaining and Preprocessing the Sensor Data

As presented in Chapter 3, where we show how we integrate this sensor into a textile icon, the exact sensor we chose is the ICM-20948 from InvenSense[1]. Here, we want to concentrate on the software that deals with its data. We divide this Chapter into two sections: Firstly, how we retrieve data from the sensor, and second, how we preprocess it with a sensor fusion algorithm.

4.1 Communication with the Sensor

We fetch data from the sensor at 100 Hz.

Since the IMU combines three sensors — an accelerometer, a gyroscope, and a magnetometer — we can treat it as three separate sensors. Each of these three sub-sensors is measured on three axes, resulting in nine degrees of freedom for the whole IMU or nine values to retrieve each time it updates. We fetch data from the sensor at 100 Hz, which is limited by the maximum data rate of the slowest sensor, the

[1] `https://invensense.tdk.com/products/motion-tracking/9-axis/icm-20948/` as of 09.2024

magnetometer. That means we receive a new measurement from each sub-sensor every hundredth of a second.

The measuring ranges of the sensor were not changed.

While the sensor allows the measuring ranges of each partial sensor to be adjusted, we did not notice any significant changes in its accuracy for our use case, so we did not modify any of the settings. All ranges are, therefore, set to the biggest value possible by default. However, we adjusted the speed of the I^2C bus connection to 400 kHz, the fastest the sensor supports, to ensure problem-free communication between the sensor and the microcontroller. Please note that this setting does not change anything about fetching new measurements at 100 Hz.

The microcontroller should have a decent processing speed and some non-volatile memory.

In general, the sensor can be used with any microcontroller that can communicate via serial with a PC and via I^2C with the sensor, but there is a requirement for processing speed. Since the sensor and the microcontroller communicate at a frequency of 100 Hz and the sensor fusion algorithm needs to be executed for each new batch of data (more on that in the next chapter), we concluded that slow microcontrollers like the Arduino Uno are not suitable. Also, the existence of FLASH, EEPROM, or any comparable non-volatile memory helps save the sensor's calibration information. The exact microcontroller we use is the NodeMCU ESP32[2], but any comparable will work.

We use several libraries to obtain and process data from the sensor.

Adafruit provides the *Adafruit_Sensor_Lab* library to access sensors with a microcontroller effortlessly. With the help of *Adafruit_Unified_Sensor*, it generalizes all kinds of different sensor models. Unfortunately, *Adafruit_Sensor_Lab* does not support the ICM-20948. We, therefore, modified the library by adding the ICM20948 with the help of *Adafruit_ICM20X*, a library for our sensor model, and removing all other sensors from the library to minimize the code overhead and save storage space on the microcontroller. This allowed us to use the *Adafruit_Sensor_Calibration* to calibrate our sensor and *Adafruit_AHRS* for sensor fusion. The other libraries we use are *Adafruit_BusIO* for I^2C support and the *Arduino* library.

[2] `https://joy-it.net/de/products/SBC-NodeMCU-ESP32` as of 09.2024

Magnetic calibration of the sensor is critical because it can not be performed in runtime but needs to be performed once before use. Following Adafruit's guide[3], we used the MotionCal software[4] to obtain the measurements needed. While the software can send them directly to the microcontroller, we have had some problems with this feature. Instead, we copy the measurements from the MotionCal software into a script and execute it once to write the calibration information to the local EEPROM storage. We save it onto non-volatile memory so the microcontroller can lose power and the calibration is still available. Also, the *Adafruit_Sensor_Calibration* library assumes this information is saved on FLASH or EEPROM memory. We use the *sensor_calibration_read* example script to check if the calibration was written correctly to the memory.

The sensor requires magnetic calibration with the MotionCal software.

4.2 Sensor Fusion

The data we receive from the sensor should be preprocessed before using it for classification since it is noisy and has high dimensionality. We use a sensor fusion algorithm to combine all nine values each update of the sensor provides. It combines the raw data of sensors into more useful, cleaned data. We chose the Madgwick filter to combine the measurements into an absolute rotation. The icon's rotation is the only measurement needed to decide which gestures the user performs since folding and twisting are both only rotations of the icon around different axes. The filter also provides data regarding the sensor's movement, such as gravity or linear acceleration, that we do not need for our prototype but could be helpful in detecting further gestures unrelated to rotation. We used the implementation from the *Adafruit_AHRS* library. Even though this library provides two more sensor fusion algorithms, supposedly one more accurate and one less accurate than Madgwick, we did not choose the most accurate algorithm because, in our testing, it showed more drift in its calculations than the Madgwick

The Madgwick sensor fusion algorithm combines the sensor data into an absolute rotation.

[3] `https://learn.adafruit.com/how-to-fuse-motion-sensor-data-into-ahrs-orientation-euler-quaternions/magnetic-calibration-with-motioncal` as of 09.2024

[4] `https://www.pjrc.com/store/prop_shield.html` as of 09.2024

We execute the algorithm at 100 Hz.

filter. This algorithm needs to run at 100 Hz since it must be executed for each new batch of sensor data. Of course, one could lower the frequency, for example, to reduce the computational load on the microcontroller, but that would most likely lead to a loss of accuracy.

The filter provides the rotation as a triple of yaw, pitch, and roll angles or as a quaternion. The yaw, pitch, and roll system describes an object's orientation by giving three angles corresponding to rotations around three fixed axes. However, this variant of Euler angles is susceptible to the problem of gimbal lock. When two axes align, which could easily happen when the icon is rotated by 90 degrees around one axis, they become locked. From then on, they behave exactly the same, effectively downgrading the system to two dimensions, as Dam et al. [1998] described.

We chose quaternions to represent the icon's rotation.

Quaternions, on the other hand, do not suffer from the problem of gimbal lock. We, therefore, choose quaternions to represent the icon's rotation, giving us four numbers available each time we run the sensor fusion algorithm.

Excursus: *Rotation Quaternions*

Rotation Quaternions:
Euler's rotation theorem states that every rotation can be represented as a single rotation by some angle about some axis (the Euler axis) going through a fixed point, called the axis-angle representation. Quaternions extend the complex numbers and are generally of the form $a + bi + cj + dk$, where a, b, c, d are real numbers and i, j, k are the three basis elements. The axis-angle representation of a rotation, consisting of a three-dimensional unit vector for the Euler axis and an angle, can be represented by unit quaternions, that means by quaternions with a norm of 1. Because of their numerical properties and compactness, quaternions are a robust, effective, and common way of representing a rotation compared to matrices. See Dam et al. [1998] for more information.

The microcontroller sends the absolute rotation to the computer at 10 Hz.

The microcontroller sends the newest quaternion to a connected computer via USB serial communication with a frequency of 10 Hz. This frequency proved sufficient for the classification, which we will elaborate on in the next chapter. The sensor fusion algorithm should still run at 100 Hz, even if we only use every tenth output since its outputs are based on all previous observations. So, lowering the algorithm's frequency would negatively influence its accuracy. The algorithm always corrects its previous rotation estimate, which is also why the outputs in the first approximately ten seconds are very inaccurate and should be ignored: The algorithm still has to figure out the initial rotation. We found that it works the fastest if the sensor does not move in that time. Since the sensor and the microcontroller do not use much power, the computer can provide the power via USB. All of the different scripts for the microcontroller can be found in our Git repository[5].

[5] `https://git.rwth-aachen.de/i10/thesis/thesis-julian-wallerius-grablets`

Chapter 5

Classification

Supplied with a constant flow of orientations of the Grablet, we now want to detect which gestures were performed. We experimented with different systems to do the classification. For all the models, we would record the last few orientations of the Grablet and use them as input. For example, for prototyping, we converted the orientation into multiple two-dimensional images and used the $1 Recognizer by Wobbrock et al. [2007] to find a matching prerecorded gesture. In a similar way, we tried using dynamic time warping (DTW). DTW is a technique to find a mapping between two sequences and obtain a measure of how similar they are. We used it to map the recorded series of orientations to multiple prerecorded ones, each representing one gesture. We would then receive a measure of how close they are and choose the one with the highest similarity. These two systems do not require much training data but use only one template for each gesture. However, that also means that it is very hard to have multiple, slightly different templates for each gesture, considering that users perform the interaction differently from each other.

We experimented with the $1 Recognizer and DTW for gesture classification.

In the end, we chose a neural network for the classification task. It allows for a lot of configuration and can be trained with a wide range of data that can include some deviation in the performance of the gesture. The only downside is that we now must collect a set of training data. The model classifies the input sequence into one of the six ges-

We chose a neural network.

A neural network requires a training data set.

tures of folding right/left/front/back and twisting clockwise/anticlockwise or a resting class. We added the resting class to give the model the possibility to clearly classify the input, even when no gesture is currently performed. Without such a class, the model would classify the input, when the Grablet does not move, into the gesture class that is closest to it. While we could rely on none of the classes being close enough to the resting input for the model to make a confident classification, adding resting solves the problem in what we believe to be a more robust manner. For implementation, we used the Keras API[1] with TensorFlow. All of the code regarding the neural network can be found in our Git repository[2].

We added a resting class for the model to predict when no gesture happens.

5.1 Structure of the Neural Network

Our model is a sequential neural network.

Our neural network is a sequential model. That means it consists of different layers, with the first layer being the input and the last layer being the output. The structure of the model can be seen in Table 5.1. In the following section, we want to explain why we chose these layers and settings. For clarity, we will not go through the layers from top to bottom but jump a bit in our explanation.

Layer Name	Output Shape	Activation Function
Input	20×4	none
Dense	20×50	relu
LSTM	100	tanh
Dropout	100	none
Dense	50	relu
Dense	7	softmax

Table 5.1: Layers of the neural network with their shapes and activation functions. *relu* stands for the Rectified linear unit activation function and *tanh* for the Hyperbolic tangent activation function.

[1] `https://keras.io/` as of 09.2024

[2] `https://git.rwth-aachen.de/i10/thesis/thesis-julian-wallerius-grablets`

The shape of the input layer comes from the length of our time series. Since we use the data of the last two seconds with 10 Hz, and a quaternion consists of four values, we have an input shape of 20×4 values. Concerning data frequency, we tried to find a balance between enough information and keeping the network's input layer small. In our testing, 10 Hz proved to transport enough information to convey the change in rotation accurately.

We have seven output nodes since we have seven different classes the model can predict. The activation function we chose is a softmax function. It converts the values it receives from the layer above to a probability distribution, where the sum of all seven values always equals 1. We can interpret the probabilities to reflect the model's confidence in the prediction.

The LSTM layer is the main component of our model.

The main component of our network is the LSTM layer. LSTM stands for long short-term memory and is a system for neural networks developed by Hochreiter [1997]. As Van Houdt et al. [2020] described, the LSTM architecture consists of memory blocks that can maintain their state over time. Therefore, LSTMs are often used for tasks like time series prediction or time series classification. We selected the hyperbolic tangent function because it is normally used as the activation function.

The dropout layer in the network randomly sets values to 0 in the training stage to prevent overfitting. Since fighting overfitting is important for our small data set, and approaches like by Karim et al. [2018], Karim et al. [2019], and Tan et al. [2019] also use this layer behind an LSTM layer for time series classification, we adopted the usage for our model. Similarly, we chose the dense layers with their respective activation functions, listed in Table 5.1, in front and behind our LSTM layer, and followed the example of Tan et al. [2019]. Dense layers are just regular, fully connected feed-forward layers.

5.2 Training the Model

We recorded us performing the gestures 280 times.

The training process is a fundamental part of a neural network. The first hurdle we encountered was the need for many data samples to train the model. To obtain these, we wrote a script to record two seconds of sensor data every five seconds. Using that script, we could record the same gesture multiple times. We recorded each gesture 40 times and tried to introduce some variation into the dataset, for example, by varying the intensity of the interaction. Overall, we had 280 data points with which we trained our model.

We used a special cross-entropy loss function.

Since the task of the model is classification, we used the *sparse categorical cross-entropy* function as our loss function. It computes the cross-entropy loss but is specialized for classification with encoding the different classes as integers. When labeling the training data, we could decide between using integers (so using 0 for resting, 1 for folding right, and so on) and a vector with seven numbers, each representing one gesture, from which exactly one would be 1 and all others 0. We chose the first option. However, except for the choice of loss function, this decision should not have influenced the model's structure or performance but was more a question of personal preference.

We used the Adam optimizer.

We applied the Adam optimizer with the default parameters for our model. It is a very common optimization that is based on stochastic gradient descent. It was first presented by Kingma and Ba [2017], who described it as computationally efficient and having little memory requirements.

The model was trained with k-fold cross-validation.

We trained the model using k-fold cross-validation with $k = 5$. This method shuffles the data and trains five separate models. For the first one, the first 20% of the data are used as validation data, and the other 80% are used for training. For the second one, the second 20% are used for validation, and so on. In the end, out of the five trained models, we chose the one with the highest accuracy. We picked that training method because it avoids overfitting, which is usually a big problem for small sets.

We trained the model for a maximum of 80 iterations (epochs). However, we stopped the training process as soon as there was no improvement in minimizing the loss function for five epochs. The model never reached the 80th iteration in training but was always stopped by this condition. By that, we avoided unnecessary training but also never stopped training prematurely when improvement still was possible.

Training lasted a maximum of 80 epochs but was always stopped prematurely.

Ultimately, we converted the model and saved it as a TensorFlow Lite model. This is a more efficient and compact way of saving a smaller model like ours and also speeds up the inference. This is very important since we need to perform the inference ten times a second and additionally would like to display the output on a real-time bar chart. Without the conversion, we had trouble reaching this performance on our machine. The model achieved an accuracy of 100% for the validation data.

We converted the model to a TensorFlow Lite model to increase its speed.

5.3 Interpreting the Output

The output of the model is a probability distribution between all the different gestures and the resting state. However, this still does not give us a classification. We need to interpret the model's continuous output to detect when it detected a gesture. Instead of a continuous data stream, we want to output the type of gesture only once, as soon as it is detected, and remain silent the rest of the time. In a figurative sense, we handled the data as if it were an analog data signal.

The probability output needs to be processed further.

Firstly, we set a cutoff value at 0.8. A probability of a gesture would need to surpass this value to be registered as a predicted gesture. This ensures that the model is confident in the prediction and does predict a gesture with slightly higher probability than the others. Whether the model predicts resting or no class reaches a probability of more than 80%, is treated equally by not outputting anything.

We require a minimum probability of 80% to detect a gesture.

We also applied the concept of debouncing to avoid a single measurement triggering the detection of a gesture. To

We applied debouncing to the signal.

be accepted as a prediction, a gesture needs to be predicted two consecutive times with a probability of more than 80%. While this introduces a delay of 0.1 seconds into our prediction since we need at least two measurements, the benefits of removing noise outweigh this drawback.

We ignore the movement back to the default position after twisting.

Also, after the user has twisted the Grablet and let it go again, the Grablet would move back to its current position, triggering the detection of a twisting gesture in the opposite direction. We implemented a system that, after detecting a twist in one direction, ignores the expected twisting in the opposite direction following it. So, after twisting clockwise is detected, the subsequent detection of anticlockwise twisting is ignored, until the resting state is reached again, and vice versa.

5.4 Attempts to Understand and Improve the Performance

We had some additional ideas to improve the model's performance. Unfortunately, of the two we implemented, none had a significant positive effect, so we did not make use of them in our final prototype. However, it might be interesting to investigate these approaches further for future work, so we would like to present them anyway.

A discrepancy between the training data and the continuous use might explain the model's behavior.

While the accuracy the model achieved during training is a good measure of how well the model was able to pick up trends in the training data, it is necessary, especially for our case of continuous classification, to test the model in a more realistic scenario that is hard to capture in one number. When testing the model, we found that it can usually pick up the gesture we performed. That means if we, for example, fold the Grablet right and observe the stream of outputs of the model, we will often see a peak in the probability of folding right. However, this peak is frequently accompanied by some peaks for other gestures. For example, we noticed that sometimes, after we performed a folding gesture and the output reflected this gesture, letting go of the Grablet would lead to the detection of a twisting gesture. This may be related to how we train our model compared to how we use it, which requires some further explanation.

Let us look at a duration of six seconds, so 60 orientations we call x_0 to x_{59}. In the middle two seconds of these, so from x_{20} to x_{39}, the user performs the folding right interaction with the Grablet. The inputs to the model can be understood as a sliding window of size 20 over this range of orientations. In the beginning, the model receives the orientations of x_0 to x_{19}. These should be similar to the training data points of the class resting and should, therefore, be classified as such. The next two seconds of input to the model are x_{0+i} to x_{19+i} for $1 \leq i \leq 20$. Here, the model receives a time series of orientations that consists of a "resting part" at the start of the window and the beginning of the folding right gesture at the end. It would be optimal if the model classifies this input most of these times as resting. Only when nearly all points of the gesture are in the input window should it start classifying them as folding right. This is where a potential problem arises since the model was never trained on such data. The same thing happens when we consider the two seconds after that, where the input is x_{20+i} to x_{39+i} for $1 \leq i \leq 20$, only that now the data points regarding the gesture slowly get replaced by points that are similar to resting. Or, even worse, the user lets go of the Grablet, and it moves back to its normal position. Now, the model is presented with a part of the gesture, a small pause, and a movement it has never seen before.
Since the interaction does not always fill the whole two seconds, this is an oversimplified example, but it nevertheless shows a problem that we believe to be responsible for some of the errors in realistic use. It can be fixed by training the model in a different way. Instead of only providing the model with the full gestures, we would need to record data in a more realistic setting and label each step the sliding window takes over this longer recording. This set of labeled data points can then be used as training data. We imagine this to be a very tedious task, especially when still aiming at getting multiple instances of all gestures into the training data set. Alternative solutions to this problem should be the focus of further work on this model.

We tried transforming the data before feeding it to the model.

An aspect of the model we experimented with was preprocessing the data even further before giving them as input to the model. For example, we converted the rotation quaternions to only represent the change between two consecutive measurements. For orientations x_0 to x_{20} we created y_0 to y_{19}, where y_i represents the rotation needed to get from x_i to x_{i+1}. We then used these "delta-quaternions" to train and test a new neural network. However, we could not see any improvement in the model's performance, so we chose not to apply any preprocessing function to the data. Nevertheless, this is worth exploring further, potentially simplifying the classification task for the model.

We tried to enlarge the training set artificially.

Another thing we tested was to enlarge our dataset by multiplying and slightly modifying our recorded data points. Since the orientation on the z-axis during the interaction, so whether the Grablet is facing south, north, or anything in between, is irrelevant for the gesture classification, we wanted to copy each sample we recorded and multiply it by rotating it around the z-axis. To be more precise, for a recorded data point A, we would compute the data points A_1 to A_{120}, where A_i is rotated by $3 \cdot i$ degrees around the z-axis. We imagined that by using this data, we could train the model to ignore the initial orientation around the z-axis instead of hoping that recording the training samples in multiple directions would teach the model that this is irrelevant for gesture classification. We then trained our model on this much more extensive dataset of 33,600 samples. It achieved an accuracy of 99.00% for the validation data.
The result with this dataset, however, was not as expected. The model did not become more accurate but became more confident in its predictions. So, if one gesture's probability was higher than all others, it was most likely at nearly 100%. However, its classification jumped very quickly between different gestures, resulting in far too many gestures being predicted. It is worth investigating why the model's accuracy in the testing environment did not improve, as this would provide us with many more data points.

Chapter 6

Evaluation

To evaluate the Grablet, we conducted a user study. On the one hand, the study aimed to get an impression of how users like the fold and twist interactions with the Grablet. On the other hand, we wanted to record the sensor data to evaluate our current machine-learning model and use it to train a better one.

6.1 User Study Procedure

The main idea of our study was to have the participants perform all six gestures five times, assess how they liked the interaction, and record the sensor data. For that, we sat the participant in front of a screen and the Grablet prototype we had fixed to the table. The screen was used to tell the participant which gesture to perform next with the Grablet and when to start. The recording of the sensor data and the text on the screen was controlled from the instructor's laptop. The study setup can be seen in Figure 6.1.

We explained the gestures to the participants and let them familiarize themselves with the Grablet.

After explaining the study's purpose and procedure to the participant, we introduced them to the different gestures. We did not interact with the Grablet when explaining the gestures since we also wanted to observe how they grab and interact with it in the study. Before the 30 repetitions

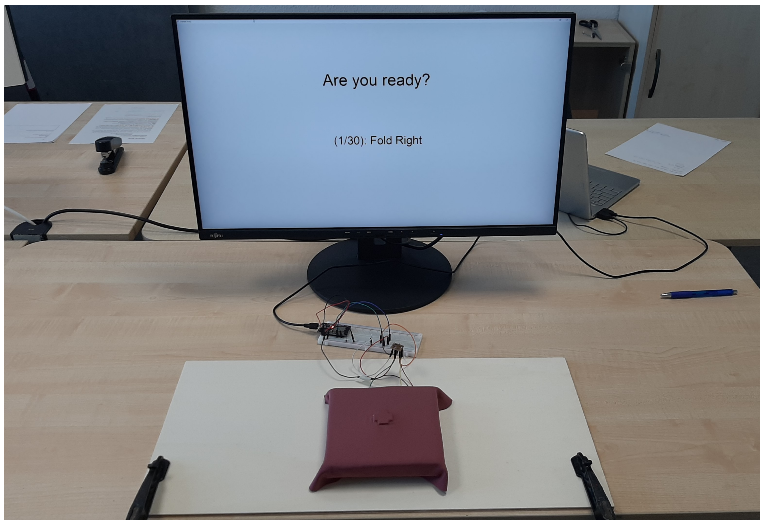


Figure 6.1: The study setup: The Grablet prototype is fixed onto the table before a screen, showing the next gesture the participant is asked to perform.

started, we asked them to perform each gesture at least three times to familiarize themselves with the Grablet and how it behaves when interacting. We also offered the participant more time to try the interface to minimize the learning effect during the 30 recorded gestures. Additionally, this was an opportunity to correct mistakes since some participants misunderstood the gestures we had explained beforehand. However, we were careful not to interfere much with how they grabbed the icon. Only when the participant did not grab the icon at all but just tapped on it, we suggested using at least two fingers to grasp the icon. As soon as the participant confirmed they felt confident interacting with the Grablet and we knew they had correctly understood the gestures and the task, we started with recording the 30 gesture performances.

We tried not to interfere with the execution of the gestures too much.

After the participant had confirmed that they were ready for the next gesture, a countdown from three started on the screen. When the countdown hit zero, the word *Start* appeared, and the participant was instructed to perform the gesture. The sensor data for the following two seconds was recorded. A quick confirmation message was shown on the screen, and the next gesture was presented, again waiting

The screen prompted the participant with the gesture they had to perform.

for the participant's confirmation that they were ready. This was repeated 30 times, five times for each gesture, with the gestures being in a random order.

In the end, participants filled out a questionnaire.

After all interactions, we asked the participants to complete a questionnaire about their experience. It consisted of ten Likert scale statements and three open questions. The statements primarily concerned the interaction's comfort and intuitiveness, while the open question asked what the participant liked or disliked overall and if there were any further comments. The questionnaire and all other forms related to the study can be found in the Appendix A.

6.2 Results and Discussion

We conducted our study with ten participants. The age ranged from 21 to 28, with a mean of 23.4 and a standard deviation of 2.54 years. One of them identified as female, and the rest as male. Eight of our participants were of German nationality; one was from Turkey and one from India. Only one person was left-handed; all the others were right-handed. While five people had much experience with textile interfaces, three had some, and two had no experience. Eight participants had a computer science background, and two came from other STEM fields.

We split our findings into two parts. The first part analyses the overall interaction, the qualitative and quantitative measures from the questionnaire, and the observations made while the participants performed the gestures. The second part evaluates our machine-learning model and trains a new one with all the obtained data. The code we used to conduct and evaluate our study can be found in our Git repository[1].

[1] `https://git.rwth-aachen.de/i10/thesis/thesis-julian-wallerius-grablets`

6.2.1 Evaluating the Interaction

The concept and look of the Grablet were mostly perceived positively.

The first and general impression of the Grablet was often a positive one. The participants with less experience with textile interfaces were especially amazed that they could interact with a simple shape on a cushion in an unexpected and new way. Also, most participants found the Grablet visually pleasing, as can be seen in Table 6.2. The measurement of willingness to use a Grablet in their own home is still pretty high but we have no value to compare it to.

	Mean	**SD**
Visually pleasing	4.5	0.67
Could imagine to use	3.9	0.83

Table 6.2: Mean and Standard Deviation for Likert scale questions concerning the overall impression of the Grablet. The scale ranges from 1 to 5, where 1 indicates disagreement and 5 agreement.

Many were insecure about how strongly they could interact with the Grablet.

Many participants mentioned that they were scared to break the prototype. One participant described the Grablet as "delicate", while another found that being able to put a finger under the icon made it feel unstable. Often, during the familiarization stage, we had to encourage the participants to increase the range of motion they performed with the Grablet while still being reasonably careful, to which many reacted with surprise. One participant wished for a threshold limiting the interaction range to "safe zones" and suggested adding a non-elastic piece to the bottom of the fabric to stop too big movements. This insecurity presumably negatively influenced the comfort of all gestures.

Twisting was more comfortable than folding.

An interesting finding from the agreement statements is that people seem to find the twisting interaction more comfortable than folding. When examining the mean and standard deviation for the questions regarding comfort in Table 6.3, it is noticeable that the participants not only found grasping the icon to twist it more comfortably but also liked the overall interaction more. This is also reflected in the answers given to the open questions. For the twisting gestures, some participants remarked that grabbing the

Grablet in the corners felt quite good, while holding it on the long sides did not. We could observe that most tried multiple ways of grasping the Grablet to twist it, but as the study went on, they most often grasped at least two opposite corners of the icon. However, twisting the Grablet was not really an intuitive gesture for the participants. From all the questions we asked, it received the lowest score.

	Fold		Twist	
	Mean	**SD**	**Mean**	**SD**
Grasp Comfort	3.4	0.92	4.3	0.90
Overall Comfort	3.8	1.08	4.3	0.78
Fitting Resistance	4.1	1.14	4.2	0.98
Affordance for gesture	3.4	1.36	3.1	1.04

Table 6.3: Mean and standard deviation for Likert scale questions differentiating between twist and fold interactions. The scale ranges from 1 to 5, where 1 indicates disagreement and 5 indicates agreement.

Many participants wanted to just press the Grablet to fold it.

For folding, the interactions differed more between participants than for twisting and significantly deviated from our expectations. In the familiarization stage, six participants were observed performing the fold gestures by pressing with one finger on the side of the icon to which it should be folded. For some of those, the instructor's correction to use at least two fingers so they could really grab the Grablet led to them introducing a second finger on the opposite, "higher" side of the icon that supported the pressing finger by pulling a bit. However, this still did not really resemble a grabbing interaction since the finger on the "lower" side of the Grablet was mostly placed on the edge instead of on the side, which is not the interaction we aimed for.

For folding, the Grablet was perceived more like a D-pad than something to grasp.

Instead, the open questions and comments that some participants made during the study indicate that they perceived the Grablet, in combination with the folding gestures, more like a D-pad. With a D-pad, we refer to a control pad in plus form, commonly found on game consoles. One participant wrote about the folding interaction that "it felt like something you shouldn't do." The mean for whether the icon looks and feels like it can be folded/twisted, in other words, whether it affords the gestures, in Table 6.3

shows that folding was slightly more intuitive than twisting. The answers to this question might be biased since the participants had just interacted with this icon in the presented ways. However, the shape of the Grablet seemed to imply some interaction possibility in the four directions correctly. The participants appeared to interpret this cue differently from each other, as can be seen by the high SD compared to the other questions.

It was suggested to modify the shape of the Grablet towards a joystick.

To improve the perception of the Grablet, one participant suggested modifying it to reflect the possibility of folding better by giving it the shape of a joystick. Having the Grablet stand out significantly and giving it a bulb-like shape might increase the impression of such a joystick and, therefore, suggest moving it in ways similar to the folding we now have with the Grablet. Another suggestion was to add notches to the sides of the icon to improve the grip for the folding gestures.

Folding left was most uncomfortable.

The placement of the Grablet potentially caused inconvenience.

From the different folding directions, some participants found folding back to be the most comfortable. Folding left was described by multiple right-handed people as very uncomfortable. This may be connected to another aspect some participants addressed: Since we placed the Grablet directly in front of the participant, two participants suggested sitting a bit more sideways, having the Grablet in front of the shoulder of their dominant hand. To fold left, most participants placed their thumb on the left side and their index finger on the right side of the Grablet, requiring them to put their right arm directly in front of their body, which was uncomfortable. We need to explore the influence of the Grablet's placement relative to the user's body on comfort further.

6.2.2 Evaluating the Machine-Learning Model

We can use 290 gestures recorded in the study.

We recorded 300 data points during the study: 5 per gesture and participant, with 10 participants, comes to 50 per gesture. We needed to remove 10 points from our set because of misreadings of our sensor, the participant performing the wrong gesture, or the Grablet slipping out of their hands during the gesture. We removed folding back six times, folding right two times, and folding front and twisting clockwise each one time. The high number of removed folding back gestures originates from a hardware problem the prototype developed during the study, where folding the Grablet at a steep angle backward would lead to a connection loss to the sensor.

We tested the model with a cutoff percentage of 50% and 80%.

We tested our machine-learning model by letting it predict the gestures of each recording. In Chapter 5.3, we described that for normal use, we would use a cutoff percentage of 80%, so we require the predicted probability of a gesture to be higher than 80% for it to count as the classified gesture. However, for that value, 154 samples, so 53.10% of the dataset, were not classified because no gesture had a probability of more than 80%. Therefore, we wanted to analyze the data further and also computed the predicted gestures for a cutoff percentage of 50%. The number of samples that were not classified shrank to 20 (6.90%). Table 6.4 shows the number of not classified data points and the accuracy of our model.

	50%	**80%**
Not classified	20	154
Classified	270	136
Correctly classified	110	79
Correctly classified (% of all)	37.93%	27.24%
Correctly classified (% of all classified)	40.74%	58.09%

Table 6.4: Accuracy of the model predictions for 50% and 80% cutoff value. We also provide the accuracy only considering data points classified into a gesture class, effectively treating not classified points as non-existent.

While the model seems to perform better for 50%.

Of course, lowering the cutoff value, apart from increasing the absolute number of classified data points, also increased the number of correctly classified points. When we look at the accuracy, one might conclude that our model performs better with a lower cutoff value. However, when we ignore all the unclassified data points, the higher cutoff value leads to a higher accuracy. In other words, while lowering the cutoff value leads to more correctly classified gestures, all the additional data points that were previously not classified are now classified with low accuracy. The ges-

		Fold				Twist		Sum
		Right	Left	Front	Back	CW	ACW	
Fold	Right	26(29)	-	0(2)	-	-	-	26(31)
	Left	-	15(21)	-	-	-	-	15(21)
	Front	-	-	5(10)	-	-	-	5(10)
	Back	-	0(1)	-	2(6)	-	-	2(7)
Twist	CW	1(1)	-	1(6)	-	0(2)	1(1)	3(10)
	ACW	12(16)	-	5(17)	0(6)	3(18)	31(42)	51(99)
Resting		-	12(19)	1(9)	6(30)	14(29)	1(5)	34(92)
None		9(2)	23(9)	37(5)	36(2)	32(0)	17(2)	154(20)
Sum		48	50	49	44	49	50	290

Table 6.5: Confusion Matrix for the study data and our machine-learning model. The x-axis shows the gesture, and the y-axis shows the model's prediction. The first number refers to a cutoff rate of 80%, and the number in brackets to a cutoff rate of 50%. When no gesture had a higher probability than 80% (50%), it is listed as *None*. Correct predictions are underlined. For clarity, instead of writing 0(0) we only wrote -.

tures predicted with a probability of at least 80% were correct in 58.09% of the cases, a significantly higher accuracy than the same metric for a cutoff value of 50%. This result was expected since the probability the model assigns to each gesture resembles its confidence in the classification. However, it also puts the low initial accuracy of 27.24% into perspective. Therefore, simply lowering the cutoff value to get more classified gestures should not be the solution. Instead, we should focus on increasing the confidence and overall performance of the model.

Anyway, lowering the cutoff percentage is not a good solution.

Most errors are the model predicting twisting anticlockwise, resting, or not classifying it as any gesture.

To gain more insight into the classification errors the model made, we can compare the predictions to the actual gestures and create a confusion matrix. The matrix for both cutoff values can be seen in Table 6.5. The first thing we can notice is that most errors seem to be made in one of three cases: When the model predicts twisting anticlockwise, resting, or does not classify the data point as any gesture. The last case is, as discussed, less pronounced for a cutoff value of 50%. On the other hand, when the model predicts any other class, the probability of the prediction being correct is quite high. The model seems to have a problem with preferring some classes over others. This may be a problem caused by our small training data set.

Even if we focus on the 58.09% from Table 6.4, the model performance is definitely not sufficient for actual use. Also, many were scared to break the prototype or unsure how big the interaction range was, so they performed the gestures carefully and generally not as far as we did when recording the data. This may explain the large number of gestures classified as resting. Without question, it is also a problem that only one person recorded all the training data for our model. While we did try to include variation in our data set, the high deviation of interactions we observed during the study was far more significant.

Our training data was not diverse enough.

We used the data we obtained from the study to train a new model. Since it does not matter who interacted with the Grablet when recording resting data because it is just laying flat, and we did not record any data for it during the study, we added 50 samples for the resting class from our previously recorded data set. The model was trained with a total of 330 data points and achieved an accuracy of 94.12% on the validation part of the training dataset. To answer whether this model performs better than our previous model requires further testing. On the one hand, we now have a much more diverse dataset to train our model with, which should greatly enhance its average performance when used by many different users. On the other hand, we collected only five data points per gesture and participant, which might not be enough. The evaluation of this model remains a task that needs further research.

We trained a new model with the recorded data.

Chapter 7

Limitations and Future Work

During the fabrication of our prototype and while conducting our study, we noticed some limitations of our work and some areas that offer to be studied in more detail in the future. For example, in our study, we placed the Grablet directly in front of the participants. As discussed in Chapter 6.2.1, some participants found this interaction uncomfortable. Further studies could be conducted on the influence of the body position relative to the Grablet on the comfort and accuracy of the interaction. Also, it would be beneficial to have a more realistic study setting where the Grablet is placed on the armrest of a couch and the participants control a simple smart home environment. The results from our study might also be influenced by our small and quite homogeneous group of participants. We mostly had young, male, and technically experienced participants, most of which already had at least some experience with textile interfaces. It is worth investigating if people with other demographic backgrounds interact with the Grablet differently.

We could vary the position of the Grablet.

The study results might be influenced by our small participant group.

In its current form, our prototype has some connectivity issues, and the electrical components are quite fragile. Also, as one participant noticed in the study, the wires leading to the PCB sometimes can be felt, especially when folding the Grablet. With a professionally produced PCB and thinner

There are some modifications to enhance the prototype's robustness and comfort.

wires, improving the prototype's robustness and durability should be fairly easy. Another idea we learned from the user study is that users might like feedback on how far they can safely move the Grablet. This could be solved by integrating some hidden, non-stretchable fabric into the prototype, which limits the range of motion. Giving the user confidence in interacting might positively influence the comfort of the interaction and is something worth investigating further.

We could vary the icon shape and properties of the fabric.

An interesting property of our prototype is the icon shape. We only produced a Grablet with the shape of a plus. It would be interesting to investigate how different icon shapes influence the interactions' intuitiveness and comfort. Also, a different shape might not lead to the user confusing the Grablet with a D-pad for the folding interactions, as observed in our study. We can even consider entirely different 3D shapes that stand out even more from the surface, not limiting ourselves to icons. This might be the most important research question for developing the Grablet further since the participants' misinterpretation of the interaction was the main problem of our study. A similar aspect is the type of fabric used. Our choice of fabric has certainly influenced the comfortability of the interaction. While some participants did express their liking of the chosen fabric, it is unclear how changing its elasticity and feel would influence the interaction.

The idea of a Grablet can be applied to wearables but would require much further research.

Currently, we assume that the Grablet is on an object that does not move constantly but is mostly motionless as long as there is no interaction. This greatly simplifies the process of determining which gesture was performed by not having to consider movements that are caused by the whole fabric moving rather than just the icon. While the Grablet can be used for smart home applications with this assumption, for example, on the armrest of a couch, it limits the use cases, especially by excluding wearables. We think it could be possible to train a machine-learning model to detect whether or not somebody really interacted with the Grablet, possibly by adding a touch sensor or more IMUs in different parts of the fabric. However, further research is needed on the user's interaction with a Grablet on their clothes to estimate the potential of the interface, before it makes sense to think about the technical implementation.

A Grablet that can handle a dynamic resting position would be good for more smart home applications.

Even when staying in the smart home area, we can imagine more applications for the Grablet. At the moment, the Grablet's default orientation is lying flat. We can also imagine a Grablet on the side of a couch, in a similar position to how Brauner et al. [2017] placed the control elements on their armchair. Developing a Grablet only for this initial orientation would require recording a whole new dataset for training, but it would work with a data pipeline similar to ours. However, developing a Grablet that can work in any initial orientation would be interesting since it would be one solution for every position instead of creating a new model each time. This would, most likely, lead to an extensive redesign of our data pipeline.

We could use the sensor data to detect more gestures.

We could also use more of the data the sensor and the sensor fusion algorithm provide us. Particularly interesting is the linear acceleration, primarily coming from the accelerometer. It would allow us to detect more interactions, such as when the user pulls and presses the Grablet.

We have many ideas to improve the machine-learning model.

The machine-learning model is an important part of our work and has much potential for improvement. In Chapter 6.2.2, we already pointed out the importance of gathering extensive and diverse training data for the model to ensure a better performance for many people. We presented even more ideas for improving the model's performance in Chapter 5.4. However, all of our improvement ideas for the model require accurate training data. Also, a change in the structure of the Grablet might involve changes regarding the position of the sensor or how exactly people interact with it. This is why we would prioritize researching design guidelines for the Grablet. Only after the physical prototype is nearly finalized does it make sense to start collecting more training data or testing changes in the data processing pipeline.

Refining the design should be prioritized over the model.

Chapter 8

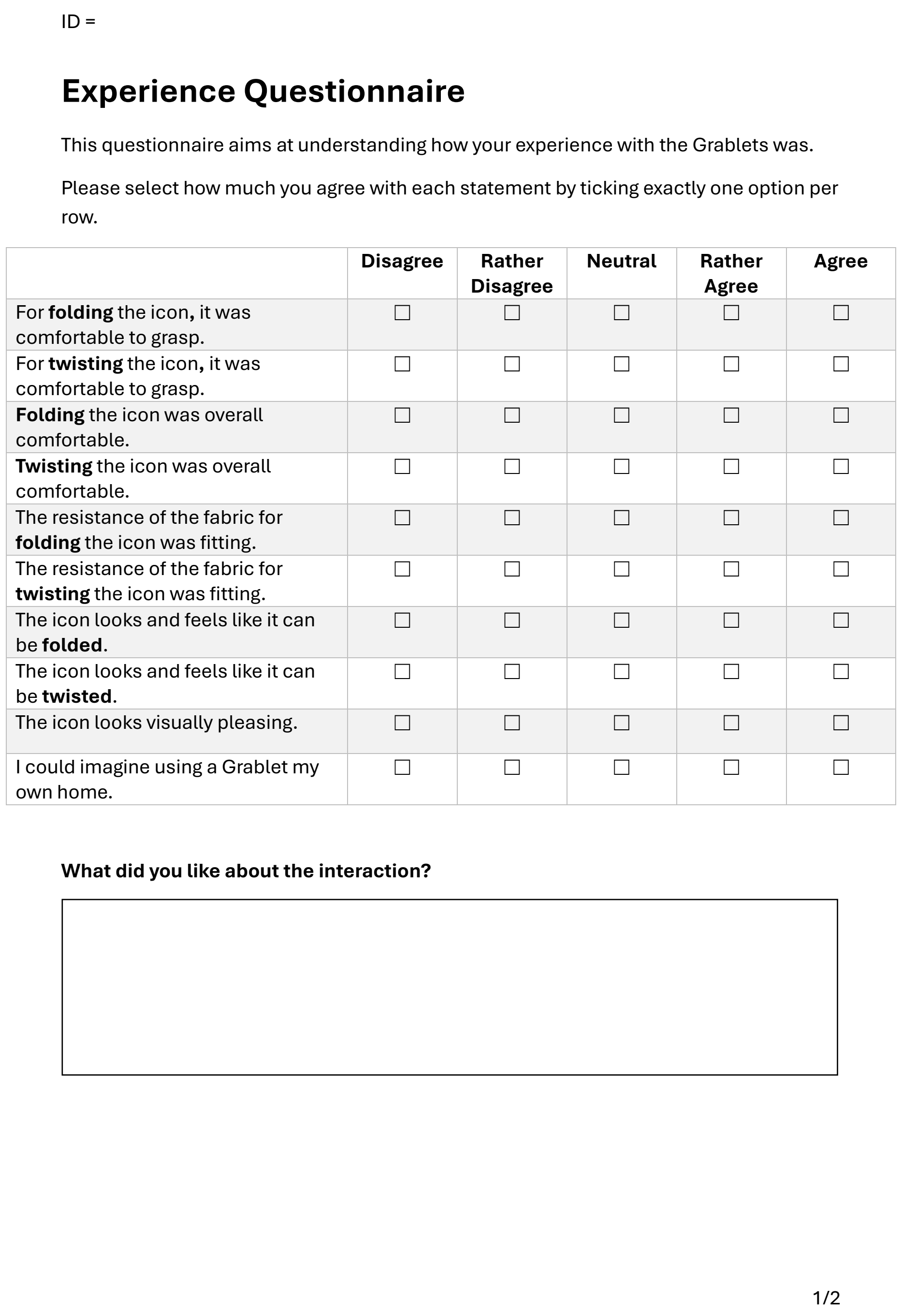
Conclusion

The goal of this thesis was to combine textile icons with interactions that make use of the fabric's deformability. For this, we developed Grablets, graspable icons embedded into the textile that can be twisted and folded. We presented our fabrication process and how we integrated an IMU into the fabric. We also described our data pipeline, which uses a neural network to detect which gesture a user performs with the Grablet. In our study, we evaluated how people liked interacting with the Grablet and how well our implementation of gesture detection worked. We found that users generally liked the Grablet, even though there was some confusion about how to perform the interaction. We acquired many ideas for refining the design of the Grablet to better reflect the interactions of folding and twisting. Our machine-learning model showed weaknesses both in the study and in a more realistic setting. We proposed some ideas about why the model's performance was insufficient and how it can be improved in the future, primarily by collecting more training data and slightly adjusting the training process. All things considered, while Grablets show promising results in enabling twist and fold interactions on textile interfaces, both the design and implementation require some future work.

Appendix A

Study Forms

On the following pages, you can find all the forms we used during our study. This includes the two-page experience questionnaire, the consent form, and the demographic questionnaire.



ID =

Experience Questionnaire

This questionnaire aims at understanding how your experience with the Grablets was.

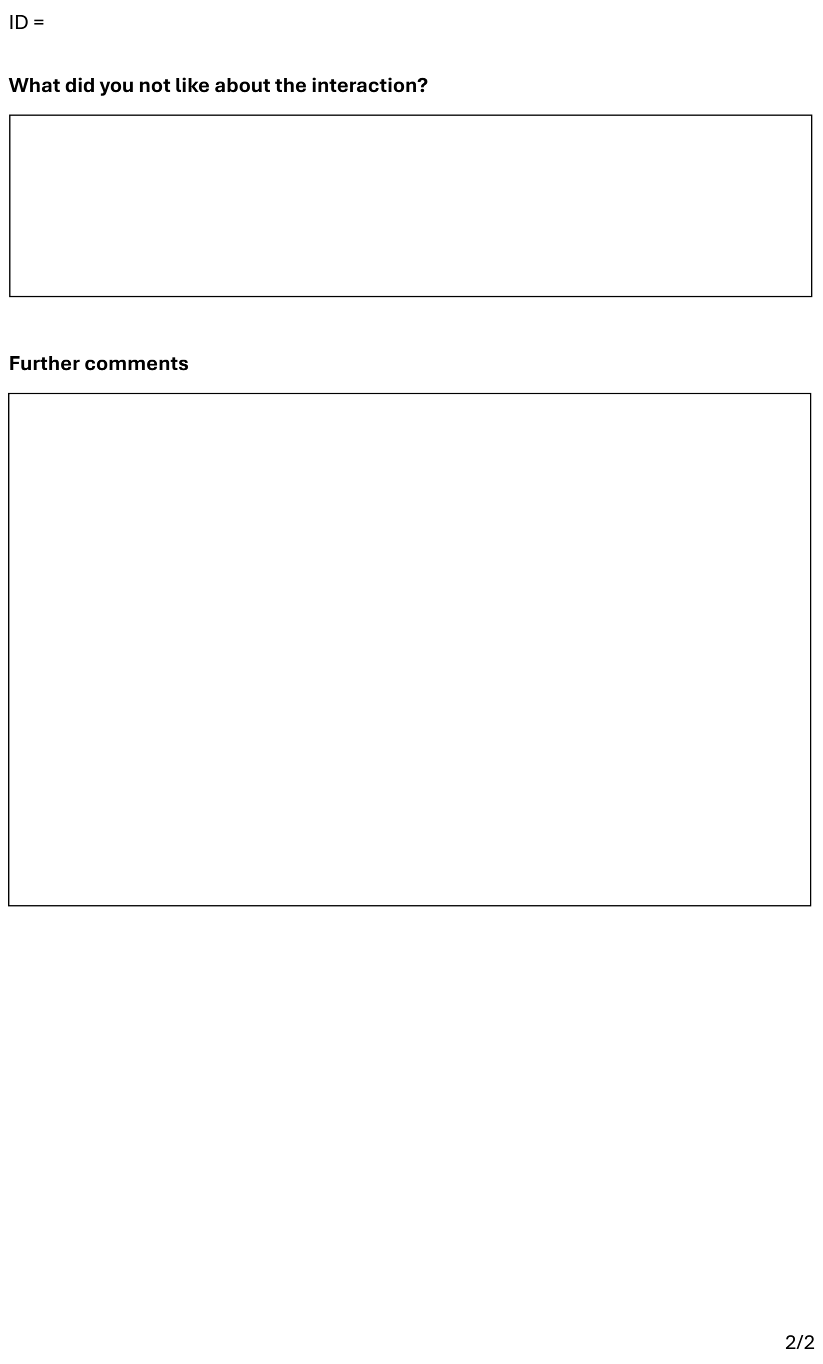
Please select how much you agree with each statement by ticking exactly one option per row.

	Disagree	Rather Disagree	Neutral	Rather Agree	Agree
For **folding** the icon**,** it was comfortable to grasp.	☐	☐	☐	☐	☐
For **twisting** the icon**,** it was comfortable to grasp.	☐	☐	☐	☐	☐
Folding the icon was overall comfortable.	☐	☐	☐	☐	☐
Twisting the icon was overall comfortable.	☐	☐	☐	☐	☐
The resistance of the fabric for **folding** the icon was fitting.	☐	☐	☐	☐	☐
The resistance of the fabric for **twisting** the icon was fitting.	☐	☐	☐	☐	☐
The icon looks and feels like it can be **folded**.	☐	☐	☐	☐	☐
The icon looks and feels like it can be **twisted**.	☐	☐	☐	☐	☐
The icon looks visually pleasing.	☐	☐	☐	☐	☐
I could imagine using a Grablet my own home.	☐	☐	☐	☐	☐

What did you like about the interaction?

1/2

Figure A.1: Study questionnaire page 1



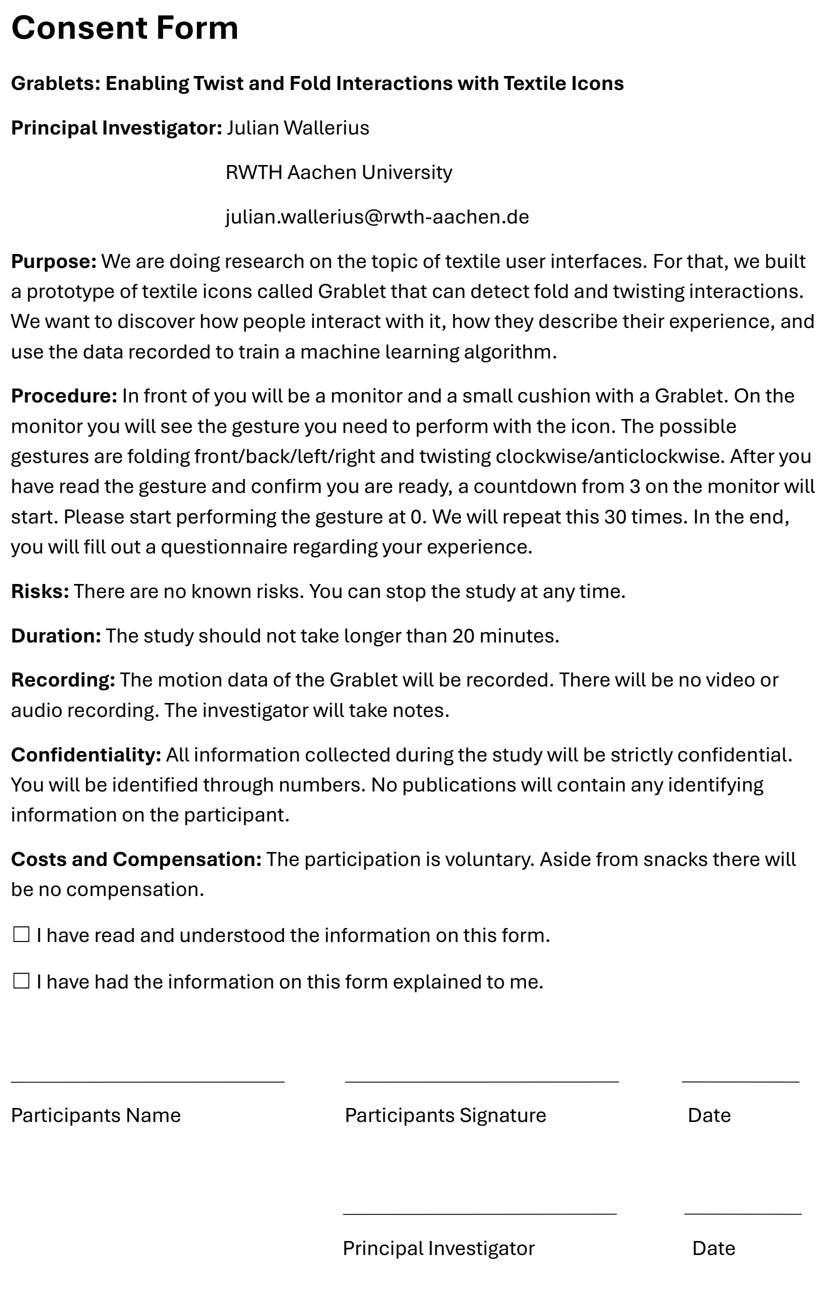
ID =

What did you not like about the interaction?

Further comments

2/2

Figure A.2: Study questionnaire page 1



Consent Form

Grablets: Enabling Twist and Fold Interactions with Textile Icons

Principal Investigator: Julian Wallerius

RWTH Aachen University

julian.wallerius@rwth-aachen.de

Purpose: We are doing research on the topic of textile user interfaces. For that, we built a prototype of textile icons called Grablet that can detect fold and twisting interactions. We want to discover how people interact with it, how they describe their experience, and use the data recorded to train a machine learning algorithm.

Procedure: In front of you will be a monitor and a small cushion with a Grablet. On the monitor you will see the gesture you need to perform with the icon. The possible gestures are folding front/back/left/right and twisting clockwise/anticlockwise. After you have read the gesture and confirm you are ready, a countdown from 3 on the monitor will start. Please start performing the gesture at 0. We will repeat this 30 times. In the end, you will fill out a questionnaire regarding your experience.

Risks: There are no known risks. You can stop the study at any time.

Duration: The study should not take longer than 20 minutes.

Recording: The motion data of the Grablet will be recorded. There will be no video or audio recording. The investigator will take notes.

Confidentiality: All information collected during the study will be strictly confidential. You will be identified through numbers. No publications will contain any identifying information on the participant.

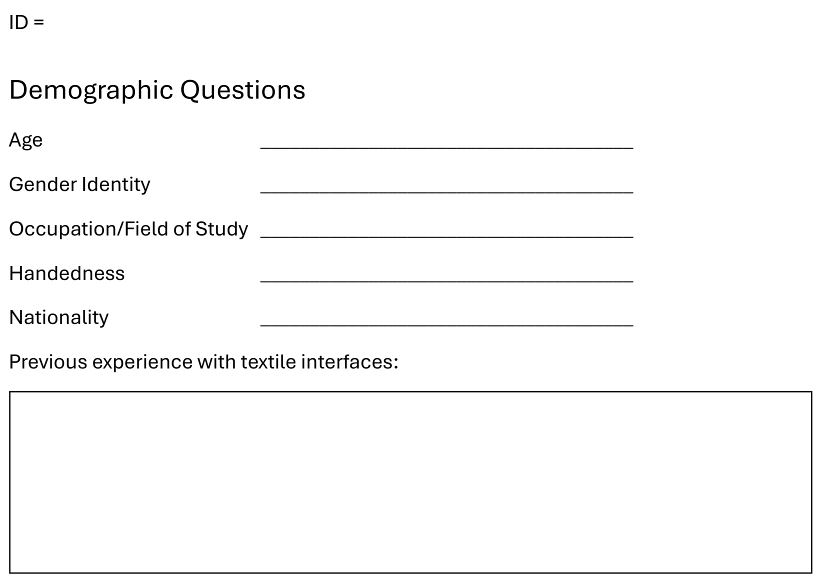
Costs and Compensation: The participation is voluntary. Aside from snacks there will be no compensation.

☐ I have read and understood the information on this form.

☐ I have had the information on this form explained to me.

Participants Name	Participants Signature	Date
	Principal Investigator	Date

Figure A.3: Informed Consent Form



ID =

Demographic Questions

Age	____________________
Gender Identity	____________________
Occupation/Field of Study	____________________
Handedness	____________________
Nationality	____________________

Previous experience with textile interfaces:

Figure A.4: Demographic Form

Bibliography

[1] Roland Aigner, Andreas Pointner, Thomas Preindl, Patrick Parzer, and Michael Haller. Embroidered Resistive Pressure Sensors: A Novel Approach for Textile Interfaces. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery. doi.org/10.1145/3313831.3376305.

[2] Philipp Brauner, Julia van Heek, Martina Ziefle, Nur Al-huda Hamdan, and Jan Borchers. Interactive FUrniTURE: Evaluation of Smart Interactive Textile Interfaces for Home Environments. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, ISS '17, page 151–160, New York, NY, USA, 2017. Association for Computing Machinery. doi.org/10.1145/3132272.3134128.

[3] Ben P. Challis and Alistair D.N. Edwards. Design principles for tactile interaction. In Stephen Brewster and Roderick Murray-Smith, editors, *Haptic Human-Computer Interaction*, pages 17–24, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[4] Erik B Dam, Martin Koch, and Martin Lillholm. *Quaternions, interpolation and animation*, volume 2. Citeseer, 1998.

[5] Guido Gioberto, James Coughlin, Kaila Bibeau, and Lucy E. Dunne. Detecting bends and fabric folds using stitched sensors. In *Proceedings of the 2013 International Symposium on Wearable Computers*, ISWC '13, page 53–56, New York, NY, USA, 2013. Association for Computing Machinery. doi.org/10.1145/2493988.2494355.

[6] Nur Al-huda Hamdan, Jeffrey R. Blum, Florian Heller, Ravi Kanth Kosuru, and Jan Borchers. Grabbing at an angle: menu selection for fabric interfaces. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, ISWC '16, page 1–7, New York, NY, USA, 2016. Association for Computing Machinery. doi.org/10.1145/2971763.2971786.

[7] Florian Heller, Stefan Ivanov, Chat Wacharamanotham, and Jan Borchers. FabriTouch: exploring flexible touch input on textiles. In *Proceedings of the*

2014 ACM International Symposium on Wearable Computers, ISWC '14, page 59–62, New York, NY, USA, 2014. Association for Computing Machinery. doi.org/10.1145/2634317.2634345.

[8] Florian Heller, Lukas Oßmann, Nur Hamdan, Philipp Brauner, Julia Van Heek, Klaus Scheulen, Christian Möllering, Laura Goßen, Rouven Witsch, Martina Ziefle, Thomas Gries, and Jan Borchers. Gardeene! Textile Controls for the Home Environment. Mensch und Computer 2016 - Tagungsband, 2016.

[9] S Hochreiter. Long Short-term Memory. *Neural Computation MIT-Press*, 1997.

[10] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access*, 6: 1662–1669, 2018. doi.org/10.1109/ACCESS.2017.2779939.

[11] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116: 237–245, 2019. doi.org/10.1016/j.neunet.2019.04.014.

[12] Thorsten Karrer, Moritz Wittenhagen, Leonhard Lichtschlag, Florian Heller, and Jan Borchers. Pinstripe: eyes-free continuous input on interactive clothing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 1313–1322, New York, NY, USA, 2011. Association for Computing Machinery. doi.org/10.1145/1978942.1979137.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017.

[14] Sara Mlakar and Michael Haller. Design Investigation of Embroidered Interactive Elements on Non-Wearable Textile Interfaces. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–10, New York, NY, USA, 2020. Association for Computing Machinery. doi.org/10.1145/3313831.3376692.

[15] Sachith Muthukumarana, Moritz Alexander Messerschmidt, Denys J.C. Matthies, Jürgen Steimle, Philipp M. Scholl, and Suranga Nanayakkara. ClothTiles: A Prototyping Platform to Fabricate Customized Actuators on Clothing using 3D Printing and Shape-Memory Alloys. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery. doi.org/10.1145/3411764.3445613.

[16] Donald Arthur Norman. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, 2013.

[17] Oliver Nowak, René Schäfer, Anke Brocker, Philipp Wacker, and Jan Borchers. Shaping Textile Sliders: An Evaluation of Form Factors and Tick Marks for

Textile Sliders. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery. `doi.org/10.1145/3491102.3517473`.

[18] Patrick Parzer, Kathrin Probst, Teo Babic, Christian Rendl, Anita Vogl, Alex Olwal, and Michael Haller. FlexTiles: A Flexible, Stretchable, Formable, Pressure-Sensitive, Tactile Input Sensor. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, page 3754–3757, New York, NY, USA, 2016. Association for Computing Machinery. `doi.org/10.1145/2851581.2890253`.

[19] Patrick Parzer, Adwait Sharma, Anita Vogl, Jürgen Steimle, Alex Olwal, and Michael Haller. SmartSleeve: Real-time Sensing of Surface and Deformation Gestures on Flexible, Interactive Textiles, using a Hybrid Gesture Detection Pipeline. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, page 565–577, New York, NY, USA, 2017. Association for Computing Machinery. `doi.org/10.1145/3126594.3126652`.

[20] E.R. Post and M. Orth. Smart fabric, or "wearable clothing". In *Digest of Papers. First International Symposium on Wearable Computers*, pages 167–168, 1997. `doi.org/10.1109/ISWC.1997.629937`.

[21] Ivan Poupyrev, Nan-Wei Gong, Shiho Fukuhara, Mustafa Emre Karagozler, Carsten Schwesig, and Karen E. Robinson. Project Jacquard: Interactive Digital Textiles at Scale. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 4216–4227, New York, NY, USA, 2016. Association for Computing Machinery. `doi.org/10.1145/2858036.2858176`.

[22] Cliff Randell, Ian Andersen, Henk Moore, Sharon Baurley, and others. Sensor sleeve: sensing affective gestures. In *Ninth International Symposium on Wearable Computers–Workshop on On-Body Sensing*, pages 117–123, 2005.

[23] J. Rekimoto. GestureWrist and GesturePad: unobtrusive wearable interaction devices. In *Proceedings Fifth International Symposium on Wearable Computers*, pages 21–27, 2001. `doi.org/10.1109/ISWC.2001.962092`.

[24] René Schäfer, Oliver Nowak, Lovis Bero Suchmann, Sören Schröder, and Jan Borchers. What's That Shape? Investigating Eyes-Free Recognition of Textile Icons. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery. `doi.org/10.1145/3544548.3580920`.

[25] Stefan Schneegass and Alexandra Voit. GestureSleeve: using touch sensitive fabrics for gestural input on the forearm for controlling smartwatches. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, ISWC '16, page 108–115, New York, NY, USA, 2016. Association for Computing Machinery. `doi.org/10.1145/2971763.2971797`.

[26] Yuri Suzuki, Kaho Kato, Naomi Furui, Daisuke Sakamoto, and Yuta Sugiura. Cushion Interface for Smart Home Control. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '20, page 467–472, New York, NY, USA, 2020. Association for Computing Machinery. `doi.org/10.1145/3374920.3374974`.

[27] Hui Xing Tan, Nway Nway Aung, Jing Tian, Matthew Chin Heng Chua, and Youheng Ou Yang. Time series classification using a modified LSTM approach from accelerometer-based data: A comparative study for gait cycle detection. *Gait Posture*, 74:128–134, 2019. `doi.org/10.1016/j.gaitpost.2019.09.007`.

[28] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8):5929–5955, Dec 2020. `doi.org/10.1007/s10462-020-09838-1`.

[29] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 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, page 159–168, New York, NY, USA, 2007. Association for Computing Machinery. `doi.org/10.1145/1294211.1294238`.

[30] Tony Wu, Shiho Fukuhara, Nicholas Gillian, Kishore Sundara-Rajan, and Ivan Poupyrev. 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, page 662–674, New York, NY, USA, 2020. Association for Computing Machinery. `doi.org/10.1145/3379337.3415886`.

[31] Guanghua Xu, Quan Wan, Wenwu Deng, Tao Guo, and Jingyuan Cheng. Smart-Sleeve: A Wearable Textile Pressure Sensor Array for Human Activity Recognition. *Sensors*, 22(5), 2022. `doi.org/10.3390/s22051702`.

Index

Typeset May 7, 2025