

# Real-Time Detection of Surface and Deformation Gestures on Flexible, Interactive Textiles Using a Hybrid Gesture Detection Pipeline

Adwait Sharma



MASTERARBEIT

eingereicht am  
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2017

© Copyright 2017 Adwait Sharma

This work is published under the conditions of the Creative Commons License *Attribution-NonCommercial-NoDerivatives 4.0 International* (CC BY-NC-ND 4.0)—see <https://creativecommons.org/licenses/by-nc-nd/4.0/>.

# Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, June 26, 2017

Adwait Sharma

# Contents

<b>Declaration</b>	<b>iii</b>
<b>Acknowledgement</b>	<b>vi</b>
<b>Abstract</b>	<b>vii</b>
<b>Kurzfassung</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Research Challenges . . . . .	1
1.3 Summary of Contributions . . . . .	2
1.3.1 Formulating Textile-Based Interactions . . . . .	2
1.3.2 SmartSleeve Hardware and Hybrid Gesture Detection Algorithm . . . . .	2
1.4 Thesis Organization . . . . .	3
<b>2 Literature</b>	<b>4</b>
2.1 Social Acceptance . . . . .	4
2.2 Limited Input Gestures on Textiles . . . . .	5
2.3 Facilitating Deformation-Based Input with 2.5D . . . . .	6
2.4 Enabling Always-Available Micro-Interactions . . . . .	7
2.5 Recognizing Gestures Seamlessly in Real-Time . . . . .	8
<b>3 Interaction Techniques</b>	<b>9</b>
3.1 Taxonomy . . . . .	9
3.2 Gesture Design . . . . .	10
3.3 Input Space and Modality . . . . .	12
<b>4 Implementation</b>	<b>14</b>
4.1 SmartSleeve . . . . .	14
4.1.1 Sensor Design . . . . .	14
4.1.2 Initial Prototype . . . . .	15
4.1.3 Unobtrusive and Robust Sewn-Based Connection . . . . .	16
4.1.4 Driver Electronics . . . . .	18
<b>5 Gesture Recognition</b>	<b>20</b>
5.1 Communication between C# and Python . . . . .	20

5.1.1	Strategy 1: ZeroMQ Messaging System . . . . .	20
5.1.2	Strategy 2: Pipe Process . . . . .	22
5.2	Machine Learning Models . . . . .	24
5.2.1	Model 1: Sensor Fusion . . . . .	24
5.2.2	Model 2: High-Dimensionality Input . . . . .	24
5.2.3	Model 3: Hybrid Approach . . . . .	29
<b>6</b>	<b>Example Applications</b>	<b>34</b>
6.1	Surface Gestures: 2D Finger and Hand motion . . . . .	34
6.2	2.5D Deformation Gestures . . . . .	36
<b>7</b>	<b>Evaluation</b>	<b>38</b>
7.1	Participants . . . . .	38
7.2	Apparatus . . . . .	38
7.3	Experiment 1: Position-Invariant Gesture Recognition . . . . .	38
7.3.1	Design . . . . .	38
7.3.2	Results . . . . .	39
7.4	Experiment 2: Testing the Heursitics Approach . . . . .	41
7.4.1	Design . . . . .	42
7.4.2	Results . . . . .	42
7.5	Technical Evaluation . . . . .	43
7.5.1	Apparatus . . . . .	43
7.5.2	Results . . . . .	43
<b>8</b>	<b>Discussion and Limitations</b>	<b>44</b>
8.1	Durability . . . . .	44
8.2	Pressure Input . . . . .	44
8.3	Three-layer Approach . . . . .	45
8.4	Use Inside Clothing . . . . .	45
8.5	Tailored Clothing . . . . .	45
8.6	Personalized Device . . . . .	45
<b>9</b>	<b>Conclusions and Future Work</b>	<b>46</b>
<b>A</b>	<b>Study Questionnaire</b>	<b>48</b>
<b>B</b>	<b>DVD Content</b>	<b>56</b>
B.1	PDF Data . . . . .	56
B.2	Video . . . . .	56
<b>References</b>		<b>57</b>
Literature . . . . .		57

# Acknowledgement

The ideas presented in this thesis are the result of multifarious discussions, email correspondences and experiments with my advisor Dr. Michael Haller, who has been instrumental in my success. With his guidance and support from day one of my masters program, I am transformed from a boy of nowhere into a researcher. I can never thank enough for the wonderful collaboration with my friends and colleagues Anita Vogl and Patrick Parzer, this work with a successful UIST submission would not have been possible without them. I am also very thankful to Christian Rendl for his enormous amount of help and teaching. Special thanks to my unofficial advisors Dr. Alex Olwal and Dr. Jürgen Steimle for providing their invaluable feedback throughout this thesis. I will be forever grateful to Dr. Ellen Yi-Luen Do for helping me find my interest in Human-Computer Interaction. I owe a debt of gratitude to the Media Interaction Lab for offering a welcoming environment and providing unique opportunities everyday. I am particularly thankful to Florian Perteneder, Joanne Leong, Kathrin Probst, Sebastian Gassler, and Teo Babic for their kindness, guidance, and feedback that fostered creativity in me. I must acknowledge the fantastic job of people involved in the Python community. Finally, I want to deeply thank my incredible friends and family for keeping me strong and grounded during all stages of my life, and supporting my interests and passions.

# Abstract

Over the last decades, there has been numerous efforts in wearable computing research to enable interactive textiles. Most work focuses, however, on integrating sensors for planar touch gestures, and thus do not fully take advantage of the flexible, deformable and tangible material properties of textile. This work introduces SmartSleeve, a deformable textile sensor, which can sense both surface gestures and deformation gestures in real-time. It expands the gesture vocabulary with a range of expressive interaction techniques, which highlight new unprecedented opportunities of advanced deformation gestures, such as, Twirl, Twist, Fold, Push and Stretch. The interaction design process, hardware implementation, and a novel non-rigid connector architecture are explained. A description is further provided about the hybrid gesture detection pipeline that uses learning-based algorithms and heuristics to enable real-time gesture detection and tracking. This presented modular architecture derives new gestures through the combination with continuous properties like pressure, location, and direction. Finally, promising results from multiple evaluations that demonstrate the real-time classification of 9 gestures with 89.5% accuracy are reported.

# Kurzfassung

In den letzten Jahrzehnten gab es zahlreiche Anstrengungen, um interaktive Textilien für den Einsatz in tragbaren Computersystemen zu entwickeln. Die meisten Arbeiten konzentrieren sich jedoch auf die Integration von Sensoren zur Erkennung von klassischen Berührungsgesten und vernachlässigen somit die flexiblen, verformbaren und haptischen Eigenschaften von Textilien. Diese Arbeit präsentiert SmartSleeve, einen verformbaren, textil-basierten Sensor, der sowohl klassische Berührungsgesten als auch komplexe Deformationsgesten in Echtzeit erkennen kann. Dieser Sensor erweitert das herkömmliche Gestenvokabular um eine Reihe von neuartigen, ausdrucksvollen Interaktionen, welche durch das Erkennen komplexer Deformationen des Textils, wie zum Beispiel Verdrehen, Falten, Ziehen oder Strecken, möglich werden. In dieser Arbeit werden das Interaktionsdesign, die Implementierung der Ausleseelektronik sowie ein neuartiges Verfahren für flexible Konnektoren beschrieben. Darüber hinaus wird die hybride Gestenerkennung vorgestellt, die maschinelle Lernverfahren und Heuristiken nutzt, um das Erkennen von Gesten in Echtzeit zu ermöglichen. Der modulare Aufbau dieser Gestenerkennung ermöglicht die Ableitung neuer Gesten durch Kombination von kontinuierlichen Parametern wie Druck, Position oder Richtung. Schließlich werden vielversprechende Ergebnisse aus mehreren Evaluierungen präsentiert, die die Echtzeit-Klassifizierung von neun Gesten bei einer Genauigkeit von 89.5% zeigen.



# Chapter 1

## Introduction

This work emphasizes the potential of deformation-based gestures for varied forms of interaction on everyday clothing to enable an expressive connection with the digital world. Furthermore, this greatly overcomes the input space constraint on current mobile devices through an eye-free and non-intrusive access.

### 1.1 Motivation

"Ultimately, computers would vanish into the background, weaving themselves into the fabric of everyday life until they are indistinguishable from it." Mark Weiser [74]

This quote is reflective of a vision to create a world where digital technologies are embedded within our real-world environment, thus making interfaces ‘transparent’. Today, a growing interest in wearable computing have overcome the key limitations inherent in traditional bulky equipments, while retaining the functionality and even with the combination of disparate features. For example, smart glasses, similar to that of Google Glass can click a picture with the wink of an eye. This evolution brings numerous explorations with unprecedented opportunities.

In this thesis, we highlight and explore the unique benefits wearable computing affords. We have focused on non-intrusive access and reducing the current state of myriad number of input devices. We close this work by speculating on how the transition to clothing-based interactions can help escape the user trapped in digital boundaries.

### 1.2 Research Challenges

With a variety of human-computer interfaces available today that brings input surface closer to the users than ever before — from traditional keyboard or mouse to miniaturized body-worn devices, including smart watches and head mounted displays. Furthermore, augmenting everyday objects and spaces (e.g., tabletops, walls, or entire floors), as well as the human body itself with interactive capabilities. Despite the ubiquitous presence such systems provide, they impose significant constraints related to size, precision, speed, and usability. Therefore, providing a single approach for eyes-free control in order to effectively and efficiently support multi-fidelity interaction is still a key challenge in an era of ubiquitous computing.

However, following this approach to enable easy-to-access and always-available input leads to the integration of technology into everyday clothing, which employs and explores new capabilities beyond smart accessories. Within the past decade, several researchers have explored and advanced various techniques on integrating sensors into textiles [5, 7, 45, 57]. Although with substantial improvements in the hardware, most of the existing work in the design space of interactive clothing still focuses on surface gestures limited to only tap and swipe techniques, and thus does not fully take advantage of the flexible, deformable and tangible material properties of textile. In other words, these unexplored physical properties of fabric for interaction design space can offer tremendous new opportunities to expand and enrich the user experiences.

### 1.3 Summary of Contributions

#### 1.3.1 Formulating Textile-Based Interactions

The goal of the work described in this thesis is to explore wearable-specific interactions, specifically driven by personal and social considerations to support varying contexts and scenarios. Unlike conventional rigid input surfaces, textiles offers an extraordinary flexibility and strength for expressive interaction techniques. As such, the basic physical properties of textiles opens a space to consider a set of advanced 2.5D deformation gestures (Twirl, Twist, Fold, Push, Bend, Grasp, Stretch, and Shake). This unique paradigm of shape deformation can enhance the existing digital interaction experience by ‘retro-fitting’ with metaphors known from the real world. Therefore, significantly expanding the design space to go beyond the current state of limited touchscreen emulation using 2D surface gestures (Tap and Swipe).

Another advantageous aspect of deformation-based interactions is to improve usability. It evokes nostalgic memories of vintage products, especially among the elderly users who mostly struggle to interact with modern interfaces. For instance, ‘twisting’ the textile affords rotational control, the analogy here is to a physical knob in radio which makes it suitable for actions with clockwise/counterclockwise motion, and can be used for applications like volume increase/decrease in virtual player. While ‘stretching’ allows the analogy of elastic input that can control the playback speed. Particularly functional, combining the pressure information with deformation provides versatile input modality to enable more degrees of freedom with expressiveness, for example, controlling fast forward speed during video streaming. Currently, such different operations are controlled by homogeneous actions, often limited to just sliding. In contrast, the proposed methodology of 2.5D gestures provides a wide range of ergonomic interactions in congruence with the end users’ mental models.

#### 1.3.2 SmartSleeve Hardware and Hybrid Gesture Detection Algorithm

This work introduces *SmartSleeve*, a textile-based sensor, which responds to both touches and deformations with pressure measurements. SmartSleeve is sewn in a sleeve of a shirt form-factor that is explicitly designed for comfortable fit and can be directly worn on the skin. It is constructed using a three-layer smart-textile composition which forms a pressure-sensing matrix. In practice, this enables both isotonic and isometric/elastic

input, as well as, state-changing interaction with integrated passive haptic feedback. SmartSleeve can be seen as an generic input sensor which is not limited to only one specific clothing.

To detect the large bandwidth of proposed interaction techniques and their feasibility, a hybrid gesture detection algorithm is designed and built through a combination of learning-based algorithm and heuristics. Foremost, this approach alleviates the tedious problem of training process for each gesture. The algorithm can derive 13 untrained gestures from 3 trained classes in a real-time environment.

Summarizing, the main contributions of this thesis in each of the three areas related to wearable-specific interactions, sensing and gesture recognition are:

1. A set of novel interaction techniques, which arise from the combination of surface gestures, deformation gestures, and continuous parameters (pressure, location, and direction).
2. A flexible, resistive-pressure textile sensor, with a novel non-rigid connector architecture using hand-sewn connection between the textile sensor and the electronics.
3. A hybrid gesture detection pipeline that uses learning-based algorithms and heuristics to enable real-time gesture detection and tracking for flexible textile sensors.
4. Two user studies that show how we recognize 9 types of gestures with approx. 89.5% recognition rate, at 30 fps.
5. A series of example applications to illustrate the immediate feasibility and potential of the proposed approach.

## 1.4 Thesis Organization

This document will firstly present a brief overview to wearable-based interactions, the challenges in creating deformable based interactions, and a summary of the contributions. Chapter 2 describes related work in 5 areas that are crucial: social acceptance, limited gestures, deformation based input, always-available micro interactions, and gesture recognition techniques. Further, the design rationales for responsive clothing are explained thoroughly in Chapter 3. The details about the hardware architecture for *SmartSleeve*, and the implementation of gesture detection algorithms are mentioned in Chapters 4 and 5 respectively. A quick overview of the example applications are demonstrated in Chapter 6. Subsequently, the evaluations are provided in Chapters 7. Chapter 8 describes the discussion and limitations. The thesis concludes with some exciting future work, followed by closing remarks in Chapter 9.

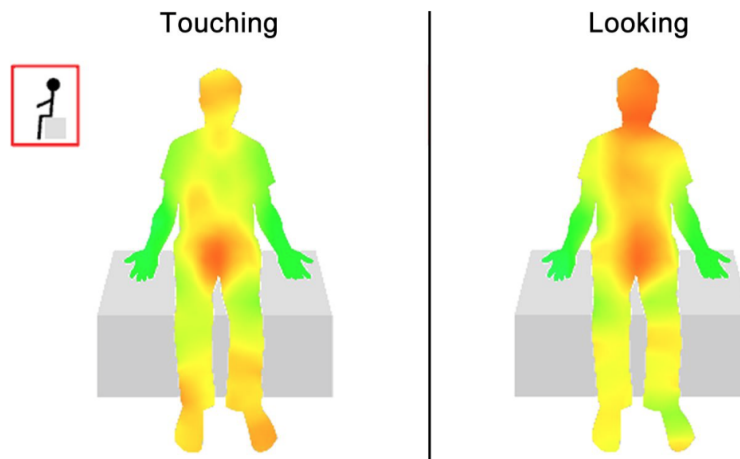
## Chapter 2

# Related Work

This chapter presents an overview that describes existing techniques on wearable interfaces. A comprehensive review that focuses on social acceptance, deformation based interactions, and techniques for gesture recognition are mentioned.

### 2.1 Social Acceptance

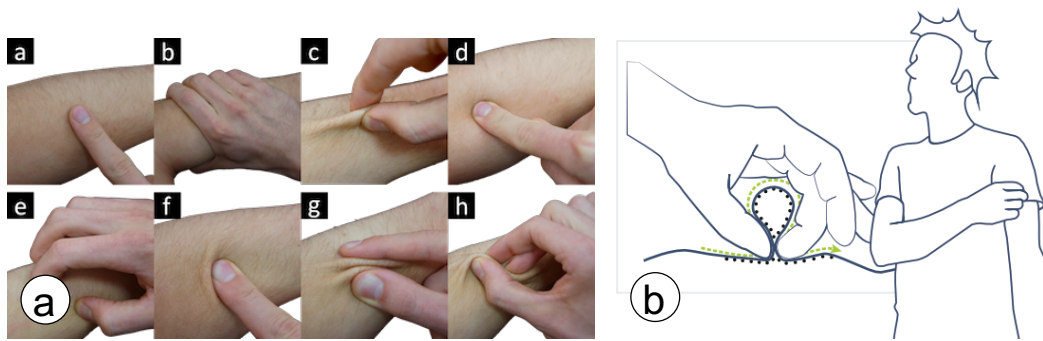
Social norms play a significant role towards one's willingness to try new things, and how readily such systems are accepted in the real world situations. Clothing choices reflects personal style and influences the interpersonal behaviors. Several designers and researchers have explored the user's perception, cultural factors and social acceptability towards the position and placement of on-body interfaces [15, 21, 40, 46, 71, 72]. The results indicate that the forearm is the most preferred region for the interaction on the arm, due to its ease of use and its comfort as shown in Figure 2.1.



**Figure 2.1:** Crowdsourced heatmap models for appropriateness in touching and looking using on-body interfaces [21].

## 2.2 Limited Input Gestures on Textiles

Several empirical studies investigated how skin or textiles can serve as gestural input surfaces. *More than touch* [72] (Figure 2.2 (a)) reported an elicitation study in a non-technological environment that shows a set of gestures including touch, grab, pull, press, scratch, shear, squeeze, and twist are preferably performed on the forearm or the hand. Lee et al. [32] explores deformation-based user gestures by using various materials like plastic, paper and elastic cloth. Bending, folding, rolling, crumpling and stretching were suggested as possible deformations. Troiano et al. [67] investigated how depth and elasticity of a display can be used to simulate deformation and provided a set of gestures including grabbing, pulling, pushing, twisting, pinching or moving.



**Figure 2.2:** (a) More Than Touch: input modalities on skin [72]. (b) Pinstripe: pinch and roll a fold of garment between fingers [30].

While previous work presented diverse gesture sets appropriate for textile input spaces, several solutions focused on specific input gestures on textiles. Touch sensitive fabrics were used for a range of gestural input on trousers [25], pockets [54] or sleeves [61]. Stitch-based solutions detect bends and folds [16, 30] by sensing interconnections between seams (Figure 2.2 (b)). Grabbing a fold at a specific angle is detected similar by using embroidered pads [19]. GestureSleeve [61] has an interesting approach for extending the input space of a smart watch to the sleeve, but only support tap and stroke gestures.

We choose to focus on a rich set of 2D touch and 2.5D deformation-based gestures on a single sleeve, which tries to combine the recent advances in the empirical studies with the current technological possibilities. Sleeve-based interfaces extends the Input Space in contrast to skin-based solutions [22, 72] by having an opening at the wrist, goes beyond conventional touch-based solutions [25, 54], and has a much large variety of interactions using the wrist [14, 69].

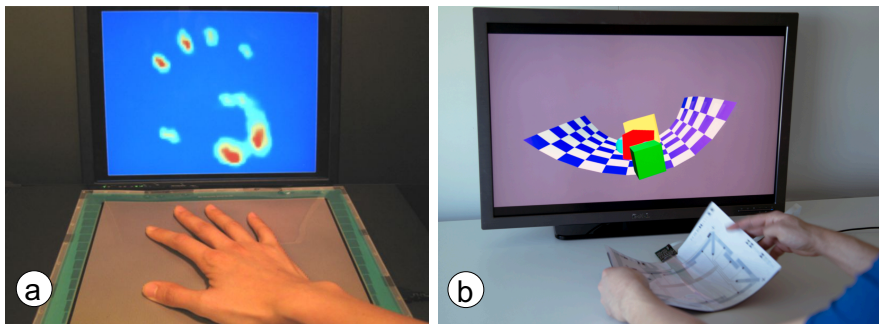
This work combines directional and pressure sensing that can deliver a wide range of novel interactions supporting additional degrees of freedom with expressiveness.

### 2.3 Facilitating Deformation-Based Input with 2.5D

Pressure-sensitive input has been a topic of interest in the HCI community for several years now, with research efforts ranging from explorations of pressure as alternative input metaphor [26, 37, 64] to the development of pressure-sensitive input devices [8, 37, 39]. To date, pressure has been used for a variety of applications such as zooming, scrolling, text entry, or widget control [38, 47, 49, 50]. A comprehensive overview of existing work in the field, can be found in [75]. However, these existing solutions are limited to a rigid form factor. In contrast, research in the domain of bendable interfaces (e.g., [17, 31, 62]) has demonstrated novel interaction techniques based on flexible sensing or input and output capabilities. Addressing this arising potential, *SmartSleeve* combines pressure-sensitive input with bending and stretching capabilities into a flexible input sensor that can form the basis for the design of more scalable, flexible, and transformable user interfaces [27].

Optical solutions presented the use of overhead cameras such as *Photoelastic Touch* [29] or structured light scanners as for *deForm* [12]. While these solutions were able to sense deformations of a flexible surface or even clay deformations on the surface, they need space for the optical tracking system. Actuated solutions such as *Relief* [33] are constructed of actuators such as electric slide potentiometers and DC motors. These approaches are more applicable for a stationary contexts instead of using them in a mobile contexts due to the fact, that these solutions need space. Ferromagnetic input solutions [56] sense on base of a matrix of sensor coils (copper wire and permanent magnets). However, the form factor is limited, the sensor coils add some weight and are more applicable for above-the-surface sensing.

Resistive solutions offer the potential of sensing deformations with a very thin form factor. *UnMousepad* [53] (Figure 2.3 (a)) is constructed of several layers (FSR surface, resistive layer, conductor, clear substrate). *FlexSense* [52] (Figure 2.3 (b)) is a thin-film sensing surface based on printed piezoelectric sensors. These solutions are already very thin by providing the ability of sensing deformations, but have a need for rigid backing.

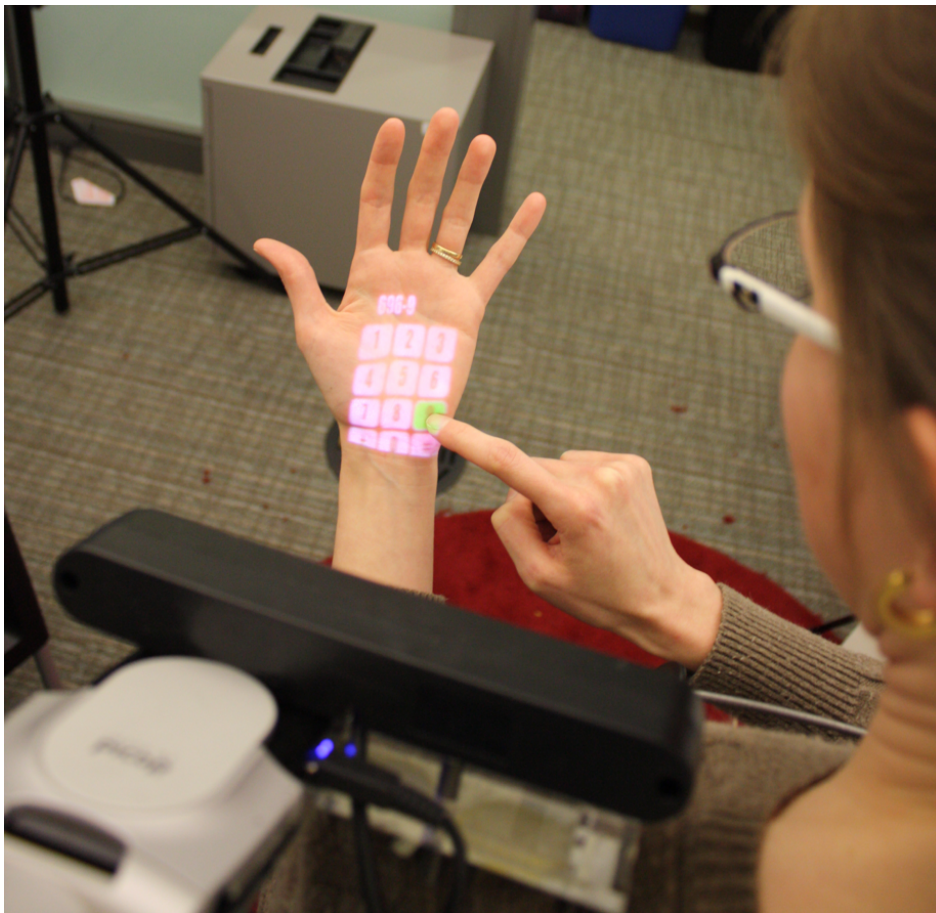


**Figure 2.3:** (a) *UnMousepad*: multi-touch force-sensing input pad [53]. (b) *FlexSense*: transparent deformable surface [52].

*SmartSleeve* is designed to be worn directly on body, and thus needs to be fully flexible and soft, while having the capability to recognize a wide range of deformations. This is achieved by its thin textile form factor.

## 2.4 Enabling Always-Available Micro-Interactions

In order to help users to perform micro-interactions, which are short-time interruptions [1], researchers have proposed a variety of ways to enable easy and fast access to mobile devices and overcome the limited interaction space on small form factor devices. Muscle input tracks the muscle tension to sense gestures [55]. Body-projected interfaces (Figure 2.4) provide visual output, which is also used for the interaction [20, 23]. Other approaches enlarge the interaction space by using sticky touch sensors [73], artificial skin [28] or enhancing the interaction space of existing devices [2, 14, 43]. While all these approaches are very diverse, they are all location variant and comes in bulky packaging of the hardware.



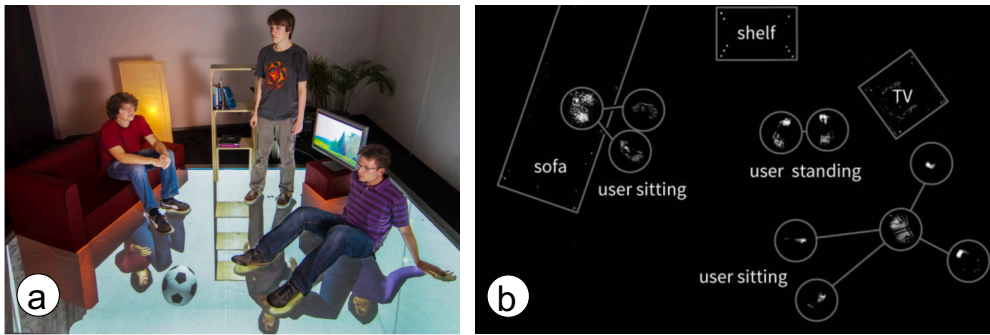
**Figure 2.4:** Shoulder-worn OmniTouch System [20].

In contrast, SmartSleeve support changes in signal at every point of sleeve, and can be worn in everyday contexts for interactions without any intrusion.

## 2.5 Recognizing Gestures Seamlessly in Real-Time

A number of pressure based sensing have explored sport and activity tracking. Most closely related to our resistive textile hardware and nature of the signal are [65, 77, 78]. Although these techniques achieve good results, recognizing various types of gestures in a single classifier requires large amounts of training which is a laborious task, and also requires extensive handcrafted features especially for temporal information which is computationally expensive as well.

The learning based gesture recognition approach only detects the trained gestures. In this paper, we propose a hybrid algorithm of combining learning-based method with heuristics that are experimentally derived. This combination enables recognition of wide variety of untrained classes with high accuracy at low computation cost, which is robust across different users and sessions. More related is the Pose Recognition mechanism in GravitySpace [3] that distinguishes five body poses on the floor as seen in Figure 2.5. We demonstrate how to extend a similar approach on clothing to derive 13 motion gestures from only three trained classes of static ones.



**Figure 2.5:** GravitySpace: (a) recognizes users and tracks their location and poses, (b) using the pressure imprints on the floor [3].

In this work, the motivation is to embrace the challenge of designing an algorithm that requires minimum training and can seamlessly run in real-time on the limited computational resources available in wearables.



## Chapter 3

# Interaction Techniques

In order to probe the design space for gestures on the sleeve, we conducted several brainstorming sessions with 9 external participants and two pilot studies (4P), including an elicitation study. Social-acceptance and preventing false positives are the essential factors considered while compiling the set of gestures. According to related work, the interaction of the dominant hand on the non-dominant upper limb is mostly preferred [72] and therefore we chosen the non-dominant hand as an input surface.

### 3.1 Taxonomy

In order to better understand the set of preliminary gestures, we manually classified each gesture along four distinct categories based on previous work [6, 76]: *Form*, *Flow*, *Property Sensed*, *Dimensionality*. Additionally, the pressure level applied while performing the gesture can add richer expressiveness. A detailed analysis about each of these categories is given in the following section:

- **Form** concerns the hand, not wearing the sleeve, which performs the gestures. Contrarily to Surface-Gestures [76], textiles provide one more dimension in the  $z$  direction. Therefore, a gesture can lead to a deformation of the textile. The hand pose can be held like *Push* or *Stretch* or can be changed *Fold / Unfold* while the deformation happens.
- **Flow** differentiates between discrete and continuous gestures. For discrete gestures like *Touch*, the response is given after completing the interaction, also continuous gestures can initiate a direct response. A few gestures can be classified as discrete or continuous depending on the use case (e.g. *Stretch*, *Twist*).
- **Property Sensed** describes the property used to sense a gesture: *position* - static position on the textile, *motion* - movement between positions, *deformation* - change of the shape of the sleeve, *pressure* - deformation into  $z$  direction, *stretch* - elastic change of the textile.
- **Dimensionality** represents the topological space of the gesture. This concerns either simple touch detection or movements on the planar surface (2D) or non-planar transformations (2.5D) changing the shape of the sleeve. A sub-categorization can be done as *on-textile*, and *with-textile* gestures. This can be used to elaborate one dimension *1D* or any combination of two dimensions *2D* or three dimensions *3D*.

Taxonomy of Sleeve-based Interactions		
<i>Form</i>	static pose	Hand pose is held in one location.
	static pose and path	Hand pose is held as hand moves.
	dynamic pose and path	Hand pose changes as hand moves.
	static pose and deformation	Hand pose is held as hand deforms the textile.
	dynamic pose and deformation	Hand pose changes as hand deforms the textile.
<i>Flow</i>	discrete	Response occurs after the user acts.
	continuous	Response occurs while the user acts.
<i>Property Sensed</i>	position	Hand / Finger touches the sleeve.
	motion	Hand / Finger moves along the sleeve.
	deformation	Hand / Finger deforms the textile.
	pressure	Hand / Finger presses down on the textile.
	stretch	Hand / Finger stretches the textile along one or multiple dimensions.
<i>Dimensionality</i>	1D	Hand / Finger moves on / deforms the textile in one direction.
	2D	Hand / Finger moves on / deforms the textile within two directions.
	3D	Hand / Finger moves on / deforms the textile within three directions.

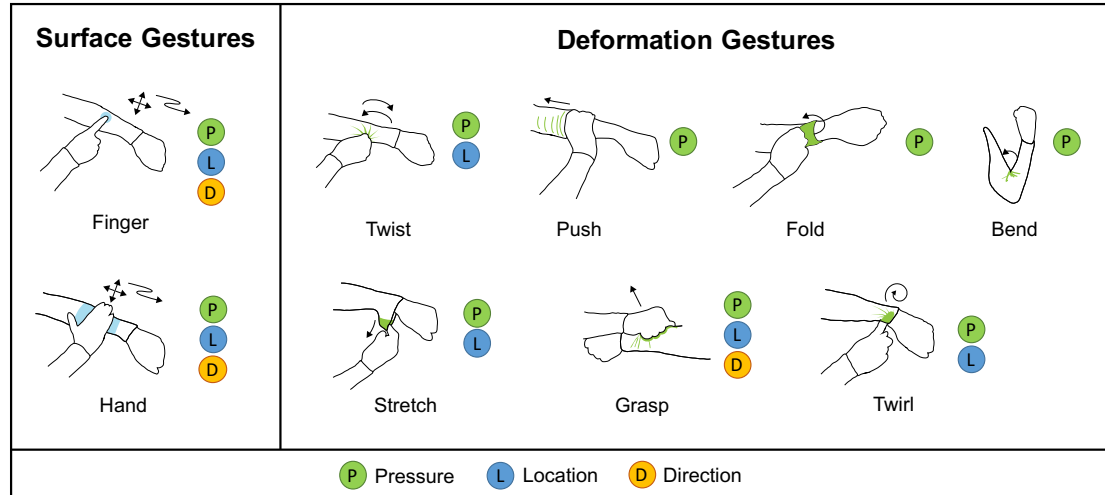
**Table 3.1:** We initially categorized interactions according to form, flow, property sensed and dimensionality.

Based on these categories, we composed a range of gestures and later classified them under two broad groups: (i) **Surface Gestures** that are planar gestures and performed on the textile, similar to conventional touch gestures, and (ii) **Deformation Gestures** (e.g., *Fold, Bend, Grasp*) that are based on deforming the textile in more dimensions.

## 3.2 Gesture Design

Concurrent with the previous research [14, 24], we initially focused on the interaction with wearables like smartwatch, smartphone, smartglasses, and earphones, since clothing is also "always-on" and provides a promising input space. On the basis of different usage scenarios (e.g., mute ringing phone, control audio volume, dismiss message, etc.) and related work about interactions with textiles [19, 30, 32, 67], we further sketched 22 types of interaction techniques after refining them with an iterative approach to create

a final set of both conventional 2D *Surface Gestures* and advanced 2.5D *Deformation Gestures* (cf. Figure 3.1 and 7.4).



**Figure 3.1:** Design space contains Surface and Deformation gesture sets (total: 22).

In contrast to rectangular loose fabrics [32] or completely fixed fabrics [67], with sleeve form factor we have one opening, which can be beneficially used for the interaction. Gestures like *stretching*, *twirling*, *pushing*, or *folding* were only made feasible due to the sleeve form factor, because in these interaction techniques the textile needs to be fixed at some sides as well as to be loose on one side. We will now discuss the *Surface Gestures* and *Deformation Gestures* in more detail.

### Surface Gestures

The most basic class we found during our early investigations are surface gestures that are performed on-textile. These are comparable to touch gestures on conventional interactive surfaces [76] and can be used for simple navigation tasks such as scrolling or swiping. Furthermore, the body shape provides a completely new property. For example, detecting a movement across the upper arm could mean a rotation around the y-axis, while the same movement across the lower arm could be recognized as a rotation around the x-axis for 3D modeling.

### Deformation Gestures

Physical deformations of the textile combined with pressure control enables novel interactions, allowing for more rich and position-independent gestures like grasping, shaking or twisting. Our participants expressed value for stretching and twisting, and related them to control zoom levels in a map application or go up/down in a navigation hierarchy. Shaking is more interesting for shuffling or can be used for swiping through images. Twisting let users scroll through lists or undo and redo actions. Therefore, a short twist returns to the last action performed in contrast to a twist, which is continuously held

scrolls back to any action performed earlier. We could also imagine combinations of these gestures or specific positions for critical commands, like twisting at the elbow for deleting a document. Additionally, gestures can occur at the end of the sleeve like pushing, stretching, folding or rolling around the finger. Pushing can be used for peeking information or zooming in. Folding requires more time and attention and is more interesting for mode switches. Twirling is very specific and can be used to fix dates like turning a knob.

Mapping uncomfortable actions like twirling can be used to perform commands, such as deleting a photo or authorizing a transaction, which require high level of consciousness [72]. Quick and rough gestures similar to rubbing or shaking offer eyes-free interaction in mobile scenarios (eg. jogging) to control applications like music player. Besides the given categories, as worn on body, the textile allows to track body movements like arm bending. Body-based gestures like arm bending are suitable for situations when the user's hands are occupied with another task, for instance, carrying shopping bags or people with impairments in their hands [70]. While twisting enables retrofitting of modern user interfaces [48], folding offers mode switching (or long-range interactions). Furthermore, combining these interaction techniques with directional and pressure sensing can deliver a wide range of novel interactions supporting additional degrees of freedom with expressiveness. Inspired by existing work, pressing and grabbing as well as twisting are most frequently performed gestures on skin-specific modalities [72].

### 3.3 Input Space and Modality

Besides the attributes described in the taxonomy, we also investigated the *input spaces* for each gesture as well as the *input modalities*. We differentiate between the *upper arm* or the *lower arm* for gestures like touch and move or deformation-based gestures such as rubbing, twisting, shaking, the *elbow* for body movements like bending and the *wrist* for interactions techniques, which require the end of the sleeve, e.g. folding, rolling up or stretching. Nevertheless, the initial focus was on the lower arm and the wrist, as an above the elbow location is closer to focus and worse from a comfort and ergonomics perspective [21]. We found four different input modalities *single finger*, *multiple finger*, *full hand* or *body part*. While the first three modalities are on the other hand, which is not wearing the sleeve, the *body part* is directly underneath the sleeve like the elbow.

The input space and modality are visually illustrated based on the participants feedback in Figure 3.2. Many gestures can be recognized at different sensor locations (e.g. forearm, elbow, upper arm). Our participants always performed *Fold*, *Twirl* and *Stretch* at the edge of a sleeve. This location offers the additional benefit of being fast to reach and easy to grasp. Gestures like *Bend* or *Twist* are limited to the physical abilities of a human.

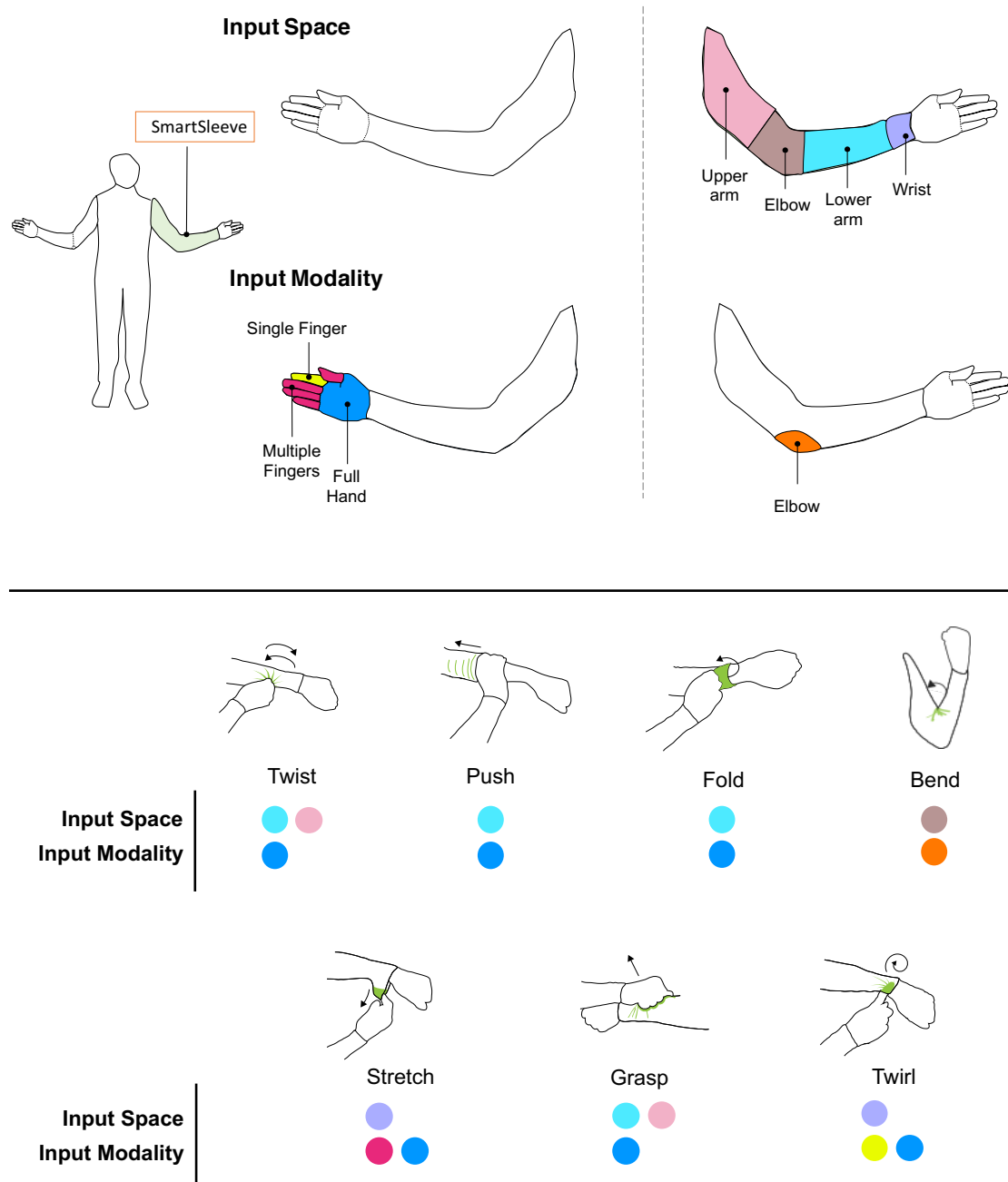


Figure 3.2: Input space and input modality for deformation gestures.

As mentioned previously, we believe that this set of gestures has tremendous potential for the *Sleeve-based interactions*, also combining the Surface-Gestures ‘on-textile’ with deformation-based gestures ‘with-textile’ could further expand the design space. The gestures shown explore the sensor classification and the sensor characteristics in the later chapters, demonstrating the flexibility and the high variety of interactions that can be detected in the real world.

## Chapter 4

# Implementation

### 4.1 SmartSleeve

SmartSleeve is a fully wearable and highly deformable textile sensor that covers a large surface, features a high amount of sensors, and offers a high pressure resolution. In this section, we present the design of the sensor and demonstrate how it can be rapidly fabricated. Table 4.1 provides an overview of the characteristics of the SmartSleeve sensor.

Parameter	Value
Force detected	50-500 g
Sample rate	100 Hz
Sensor resolution	1.66 sensor/inch
Sensor count	360 sensors
Weight in total	124 g
Length of the sleeve	40 cm
Upper arm perimeter	26.5 cm
Elbow perimeter	26.0 cm
Wrist perimeter	16 cm

**Table 4.1:** Sensor characteristics.

#### 4.1.1 Sensor Design

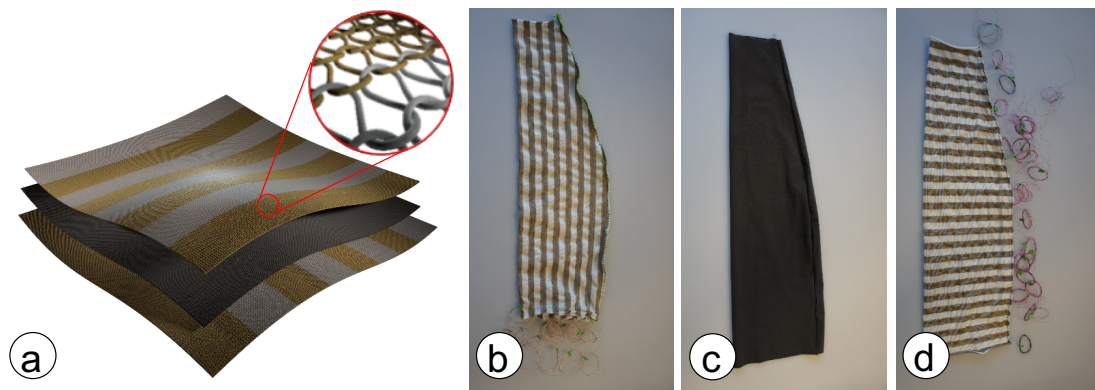
The SmartSleeve sensor builds on top of prior work [34, 42], that has introduced pressure-sensitive textile sensors that consist of three layers of fabric. We will outline how this technology can be used as clothing to enhance a wearer’s input possibilities without feeling rigid connection wires or other components added to the fabric.

All layers of SmartSleeve are equally bidirectionally stretchable and deformable. The top and bottom layers are made of Narrow Stripe Zebra fabric distributed by HITEK<sup>1</sup>, characterized by alternating strips of conductive and non-conductive fabric, see Figure

---

<sup>1</sup><https://www.hitek-ltd.co.uk/>

4.1. The strips are 9 mm wide each. The zebra-fabric layers are orthogonally aligned, to form a matrix. The middle layer consists of a pressure sensitive fabric (EeonTex™<sup>2</sup> LTT-SLPA 20 k). It has a slightly larger size to prevent the two conductive layers from shorting. Sandwiching all three layers creates a deformable and stretchable pressure-sensing matrix, which can be used to envelop complex 3D geometries. The three loose layers were stitched together along one side of the sensor to prevent the sensor grid from shifting. The sleeve constrict at the lower arm part of the sleeve, therefore an additional stretchable non-conductive fabric has been sewn lengthwise on the conductive layer, which conducts lengthwise (cf. Figure 4.1 (b)), to prevent adjacent connection lines from shorting when the layer is sewn into the sleeve.



**Figure 4.1:** The sandwich architecture of the SmartSleeve sensor (a) the bottom layer (b) and top layer (d) have conductive and non-conductive threads. In-between is the pressure-sensitive layer (c).

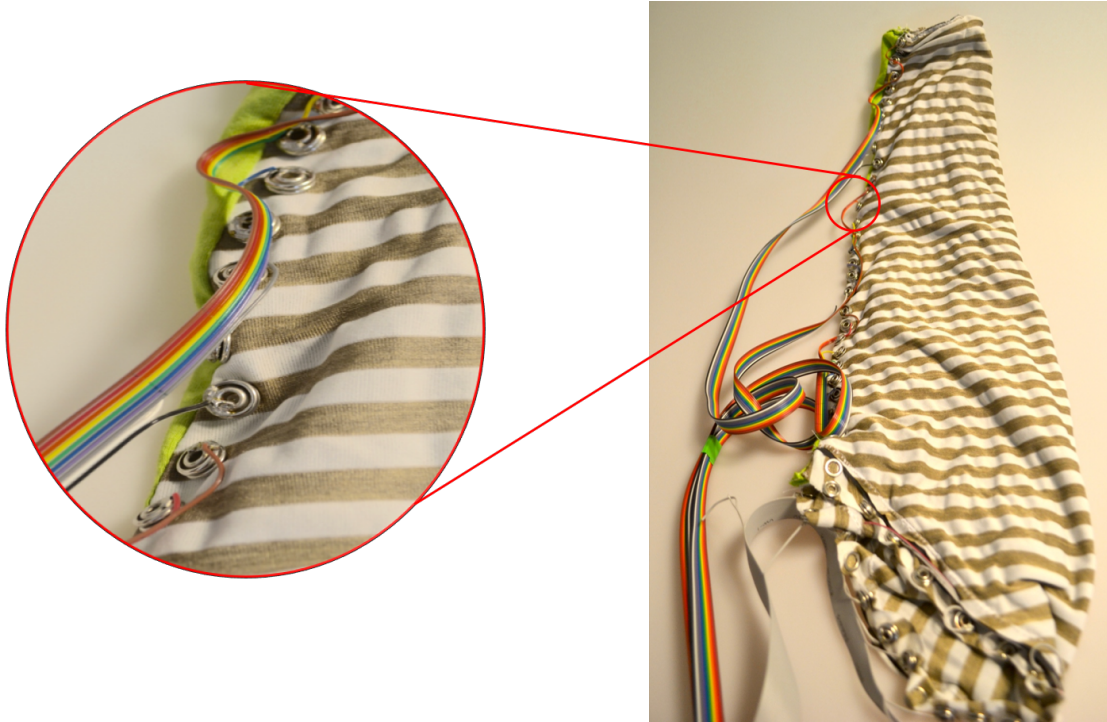
SmartSleeve is designed to cover the complete underarm and half of the upper arm. Even though this sensor technology can be easily scaled up to detect other body regions, prior work has shown that this region is most comfortable for interactions. The sleeve is designed to fit a human with a wrist perimeter size of 16 cm, elbow perimeter of 26 cm and an upper arm perimeter size of 26.5 cm. Early tests have shown that the sleeve has to fit tightly to reduce failure of short cutting adjacent wires, but not too tightly, in order to support deformation gestures. The sensor itself consists of 24 rows (around the arm) and 15 columns (lengthwise), resulting in a total of 360 pressure sensor spots with a sensor density of 1.66 sensors/square inch. SmartSleeve can be worn directly on the skin. To prevent errors from the influence of skin moisture, it was usually worn over a long-sleeved tight-fitting running shirt.

#### 4.1.2 Initial Prototype

For the first version, we followed the approach of [34, 42] that has used rigid snap buttons to connect textile with electronics (pictured in Figure 4.2) and 600 (20 × 30) sensing intersections. It is designed for the sleeve form factor. However, after initial

<sup>2</sup><http://eeonyx.com>

trials we found out that rigid connections negatively affects the comfort of the sleeve and its robustness. Therefore we explored different possibilities to connect textiles with electronics hardware.



**Figure 4.2:** The initial prototype of the SmartSleeve with metal snap buttons soldered to the ribbon cables.

#### 4.1.3 Unobtrusive and Robust Sewn-Based Connection

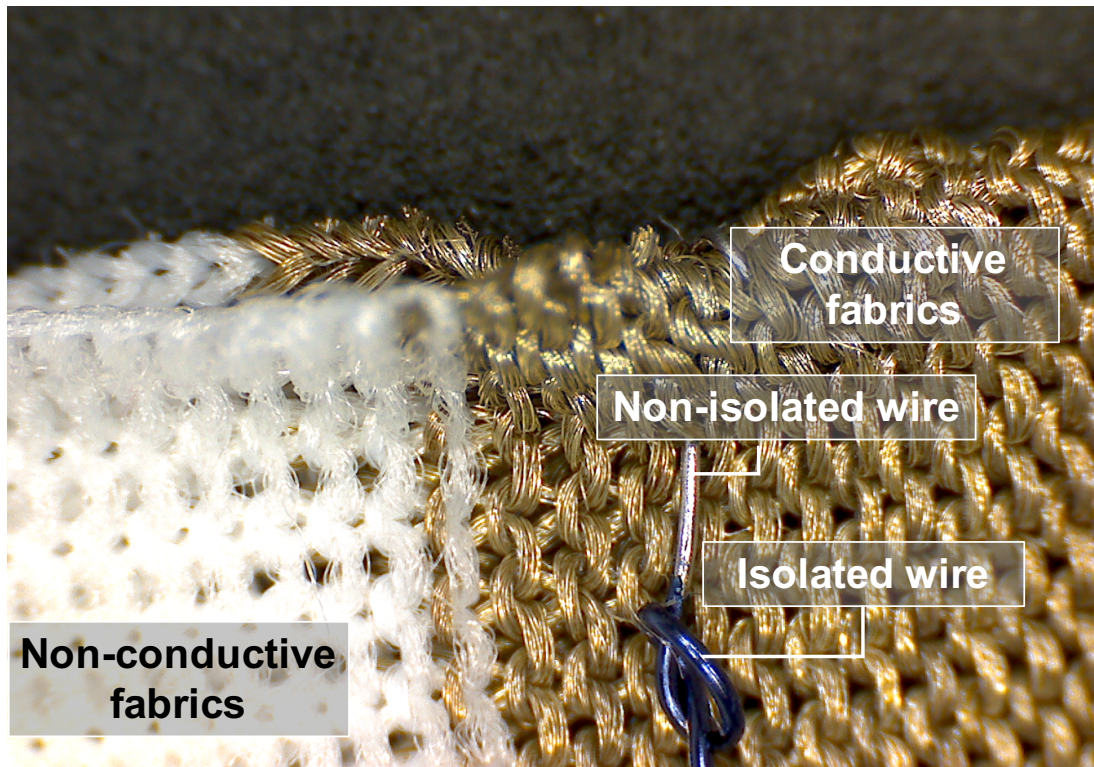
In this section, we contribute an unobtrusive and robust method to connect not-rigid, stretchable textiles with the rigid electronics. After several experiments with various materials, we learned that yarn would be the favourable connection due to its surface and shape behavior. Although many companies produce and sell conductive yarns<sup>3</sup>, very few of these yarns withstand the soldering temperature, which is required to connect the yarns with the PCB board. The main reason for that is the way how these yarns are fabricated: they either consist of multifilament yarn wrapped with metal fiber or they consist of a multifilament core yarn coated with with metal. Only the multifilament yarns wrapped with metal fibers would withstand the soldering at higher temperature properly.

A possible alternative is solderable yarn<sup>4</sup>. Although these yarns are highly conductive, they are not insulated, which makes them unsuitable for our design as they would

<sup>3</sup>[www.schoeller-wool.com](http://www.schoeller-wool.com), [www.bekaert.com](http://www.bekaert.com), [www.statex.biz](http://www.statex.biz), [www.araconfiber.com](http://www.araconfiber.com)

<sup>4</sup>*High Flex 3981 7X1 Silver*, or *High Flex 3981 Flat Braid* Karl Grimm ([www.karl-grimm.com](http://www.karl-grimm.com))





**Figure 4.3:** Hand-sewn connection between textile sensor and the electronics allow for flexible connections.

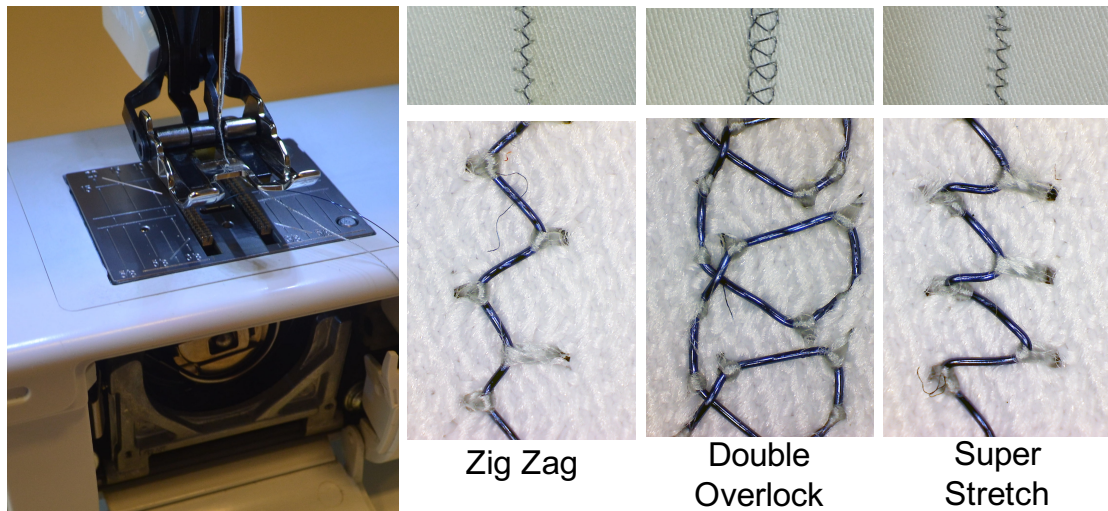
cause shortcuts. Previous research has also shown several ways to insulate conductive yarns by ‘couching’, iron-on techniques or fabric paint [4]. However, the tight sleeve needed a solution which preserves the look and feel as well as the comfort of use as much as possible.

Therefore, we opted for basing the connector on a wire with a small diameter which is conductive and insulated. During our experiments with different wires, we found the *Road Runner / Verowire* wire to be the most promising one. These copper wires are normally used for repairing or correcting printed circuit boards. The wire has a small diameter of 0.15 mm, which makes it very deformable. It is coated in solderable enamel or self fluxing polyurethane, which acts as an insulator. The coating can be removed when a high temperature is applied (400–430 °C). Hence, before sewing, we use the solder iron to remove 3 cm of the insulation.

A first method consists of handsewing the connections. A close-up of one connection is depicted in Figure 4.3. It consists of 3 cm of non-insulated wire that is affixed using a stitch to a row or a column strip of the zebra fabric. Although this stitch itself is not stretchable, it requires little area and is therefore straightforward to be sewn by hand.

In addition to the manual fabrication, we also performed initial tests with a sewing machine using different stitching modalities usable for elastic materials, including *Zig Zag*, *Double Overlock*, and *Super Stretch*, see Figure 4.4. Straight stitches or stitches

which are not adapted for elastic materials would either tear the fabric or reduce its elasticity. We found that the wire resists enough tension to be sewn with a *Zig Zag* stitch. Therefore, a non-conductive yarn was used as top thread and the wire as bobbin thread [11]. In this way, the bobbin thread can easily float on the back side of the stitch without passing through the fabric substrate when using the machine at maximum speed.



**Figure 4.4:** Fabricating the connector using a sewing machine, with *Zig Zag*, *Double Overlock*, or *Super Stretch* stitch.

These different stitching modalities have different benefits and limitations. As the wire is stiff, the equal stitch distances and the little use of yarn of the *Zig Zag* stitch preserve the comfort of use as much as possible. For the *Super Stretch* stitch, the yarn tension was raised to maintain the elasticity of the stitch. Otherwise, the wire would float in a straight line, which means that the stitch would no longer be elastic. This is due to the differences in the yarn elasticity between usual yarn (top) and wire (bobbin). Because of that, the top yarn can tear more easily. The *Double Overlock* keeps its typical pattern without making any changes regarding yarn tension. Nevertheless, we would not recommend to use this stitch for that purpose as the stitch needs much more yarn, which makes the fabric stiffer and thus reduces the comfort of use. In conclusion, we would suggest to use the typical *Zig Zag* stitch, as the pattern maintains the comfort of use, due to the simplicity of the pattern it is easily adjustable in its width and it is sewable with the predefined yarn tension and thus less vulnerable for tearing.

#### 4.1.4 Driver Electronics

SmartSleeve is based on a resistive tactile sensor. This type of sensor is subject to various sources of errors, such as crosstalk, which affect the accuracy of measurements and the gesture recognition. We evaluated different measurement principles and algorithms to determine the best solution to yield high accuracy and reduced crosstalk. First we analyzed how our system behaves with a solution without crosstalk reduction [59]. Fur-

ther, we evaluated the effects of grounding for crosstalk reduction [9], the zero potential method [63] and virtual grounding [60], the multiplexer op-amp assist approach [58], and the resistive matrix approach [35].

Measurement Principle	Average Error
Without reduction	34.5%
Grounding [9]	32.1%
Virtual Grounding [60]	42.0%
Multiplexer and Op-Amp assisted approach [58]	10.5%
Resistive Matrix Approach [35]	0.93%

**Table 4.2:** Overview of the measurement principles.

As depicted in Table 4.2, the resistive matrix approach yielded the best results and was therefore implemented in our system. The measurement electronics consist of a microcontroller, one SPDT switch, four multiplexers and four shift registers. The shift registers are daisy-chained so that they work as one big shift register. The shift register applies ground potential to the measured column while all other columns are connected to high potential. Whenever the shift register is triggered, the low level jumps to the next column. Multiplexers are connected to the row electrodes to forward single lines to the ADC. Each single sensor spot is measured separately, which means starting from the constant resistors which are mounted on the PCB to the first cells in the row and first column to all others. Then all other sensors are getting measured row by row.

## Chapter 5

# Gesture Recognition

The raw sensor data matrix from SmartSleeve is sent via a serial connection to the computer that performs various validation checks for suppression of cross-talk within the hardware measurements. To accelerate the development process without reinventing the wheel for correctness of raw input values and visualizing them from the SmartSleeve hardware, we modified the FlexTiles[42] visualization software as per the needs of our gesture recognition algorithms. However, this approach gave rise to a technical challenge of cross-language interoperability, since the serial connection is written in C#, and the gesture recognition is implemented in Python with an aim to reduce the complexity.

### 5.1 Communication between C# and Python

In order to bridge the gap between C# and Python for sensor data transmission, we developed two separate strategies as discussed below.

#### 5.1.1 Strategy 1: ZeroMQ Messaging System

The first strategy uses ZeroMQ, also known as ØMQ, 0MQ, or zmq<sup>1</sup>, which is an open source library that provides sockets to carry whole (atomic) messages asynchronously across several transports, such as in-process, inter-process, TCP, and multicast. It uses four messaging patterns for simplifying multithreading - Pipeline, Request-reply, Publish-Subscribe, and Exclusive Pair.

To receive sensor values in the Python, we implemented the Publish-Subscribe pattern (as described in Figure 5.1) where senders of messages, called publishers, and receivers as subscribers. This pattern is aimed at scalability. In other words, large amounts of data can be transmitted simultaneously to multiple subscribers, even if they exist on different devices.

With all the advantages influencing our choice for Pub-Sub pattern, there are two downsides worth mentioning in this pattern:

1. Firstly, the subscriber cannot control the rate of messages sent by the publisher, hence the messages will queue up at the end of publisher, if the subscriber cannot handle the rate of transmission.

---

<sup>1</sup>[www.zeromq.org](http://www.zeromq.org)

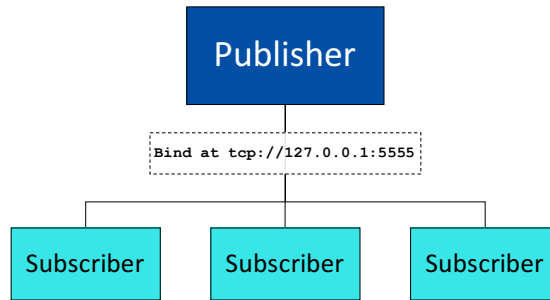


Figure 5.1: ZMQ Publish-Subscribe (Pub-Sub) Pattern.

2. And secondly, the publisher does not have any status of connection from the subscriber, both on initial setup as well as during re-connections.

#### C# Publisher

The following program (Listing 1) shows that the `sensor_data` collects data and transmits it to local publishers.

```

1  using NetMQ; //NetMQ is a native C# port of ZeroMQ
2  string[] sensor_data = new string[args.RawSensorData.Length];
3  int i = 0;
4  foreach (var item in args.RawSensorData)
5  {
6      sensor_data[i++] = item.ToString();
7  }
8  // Create a new context
9  using (var context = NetMQContext.Create())
10 using (var publisher = context.CreatePublisherSocket())
11 {
12     const string pub_address = "tcp://*:5555";
13     publisher.Bind(pub_address);
14     string message = string.Join(",", sensor_data);
15     while (true)
16     {
17         publisher.SendFrame(message); //raw sensor values as message
18         break;
19     }
20 }
  
```

Listing 1: Program.cs: C# Publisher code.

## Python Subscriber

Listing 2 script subscribes to the subscriber and prints received messages.

```

1  import zmq
2  context = zmq.Context()
3  #subscribers are created with ZMQ.SUB socket types
4  socket = context.socket(zmq.SUB)
5  socket.setsockopt(zmq.SUBSCRIBE, "")
6  socket.connect("tcp://127.0.0.1:5555")
7
8  while True:
9      sensor_data = socket.recv_string()
10     print sensor_data

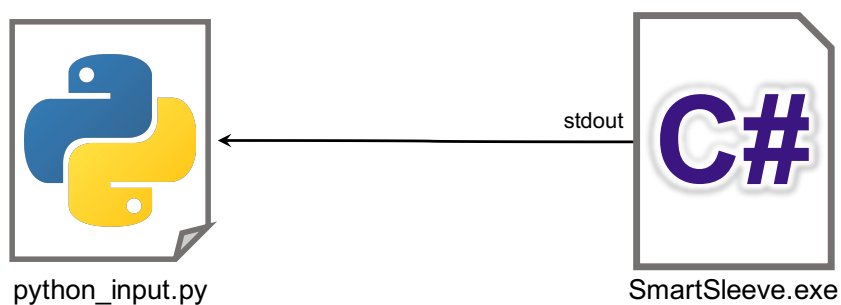
```

**Listing 2:** recv\_data.py: Python Subscriber code.

## 5.1.2 Strategy 2: Pipe Process

A simple yet ultra-fast method for interprocess communication can be attained using pipes or standard data stream. Pipes connect the output from one process as an input to the another one, without writing to a file. Such a chain of processes can be sequentially combined for more complex actions. One of the major applications of pipe is in Linux and other Unices operating systems for passing information from one program to another. There are three types of pipes: a) `stdin` allows to take input values, b) `stdout` writes the output messages, and c) `stderr` to output error messages.

To transmit data from C# app (.exe), we print the sensor data to the standard output stream (as shown in Figure 5.2). In Python, the Popen constructor takes a list of arguments to setup the new process for reading the sensor data stream.



**Figure 5.2:** Pipe process sending the standard output from one program to another.

Program.cs: C# app prints the sensor data to the standard output stream (shown in Listing 3).

```

1 using System;
2 foreach (var item in args.RawSensorData)
3 {
4     //standard output
5     Console.Write(item.ToString()+",");
6 }
7 //new line
8 Console.Write(Environment.NewLine);

```

**Listing 3:** C# code for SmartSleeve.exe.

python\_input.py: The following Python script (written in Listing 4) starts the whole process, and gets the console output of the C# application line by line as it is generated.

```

1 from subprocess import STDOUT, Popen, PIPE
2 #create subprocess with pipes
3 connect = Popen('SmartSleeve.exe', stdout = PIPE, stderr = STDOUT, shell = True)
4 while True:
5     message = connect.stdout.readline() #readline() finds \n for end of message.
6     print message #received sensor values

```

**Listing 4:** Python code for reading output from C#.

```

1 using (StreamWriter file = new StreamWriter("FileName.csv",true))
2 {
3     foreach (var item in args.RawSensorData){
4         file.Write(item + ",");} // comma as channel-value separator
5         file.Write("2"+Environment.NewLine); //gesture label
6     }

```

**Listing 5:** C# code to save raw sensor values as CSV for offline analysis.

Since, strategy 2 offers almost-zero-latency period, which is a huge advantage over the first strategy. Also, this has a lower computation cost and does not require any third-party library in contrast to the first one. Therefore, we decided to use the second strategy to implement our realtime gesture recognition pipeline.

For offline data analysis and early evaluation of models, the raw sensor values are stored in CSV (comma-separated values) files using the C# script with corresponding manually-labeled gesture names for supervised learning as described in Listing 5 . Once these data files are generated, further computation is performed in Python.

## 5.2 Machine Learning Models

Concurrent with the previous approach for C# and Python communication with different perspectives to find that one best choice amongst an array of alternatives, we built three distinct machine learning models, namely (1) Sensor Fusion, (2) High Dimensionality Input, and (3) Hybrid using Learning-based and Heuristics. The goal is to maximize the classification accuracy with minimal-training of gestures described in Figure 3.1.

We performed most of the analytical computations using SciPy, NumPy, Pandas, scikit-learn, Jupyter / IPython Python libraries. For plotting, we heavily used Matplotlib and t-SNE. Anaconda Python distribution reduced all the hustle of installing individual packages. The whole development was done on a 2015 MacBook Pro with 2.8GHz Intel Core i7 processor and 16GB RAM, running OSX and Windows (installed using Boot Camp Assistant). For Model 1 and 2 evaluation, we took a subset of 19 gesture sets from our early design space, namely no pressure, tap, touch, slide, swipe, spread, rub, bend, shake, roll up, pinch, grasp, twirl, fold, hold light, hold strong, stretch, twist left and twist right.

### 5.2.1 Model 1: Sensor Fusion

This first model is the most straightforward - we calculated the magnitude on raw sample points by combining the data from all the channels using root mean square (rms), resulting into a single vector. On a 1 second window, statistical features are computed, including mean, median, and standard deviation. Additionally, we performed fast Fourier transform (FFT) and then added a band-pass filter (5 Hz to 15 Hz) to identify the frequency domain features: power and dominant frequency. All these features are used to train a kNN classifier ( $k = 5$ ). While this approach works fairly well for classification with low-dimensional data, it almost certainly eliminates the unique property of SmartSleeve which consists high number of sensors for versatile input. As a result, this model achieved an extremely low accuracy in our early tests and takes significant amount of time and resources for pre-processing and generating predictions. Therefore, this approach seems unlikely to fulfill our need to classify all the gestures in real-time trained on a single machine-learning model, and hence caution should be taken before using this approach in actual situations.

### 5.2.2 Model 2: High-Dimensionality Input

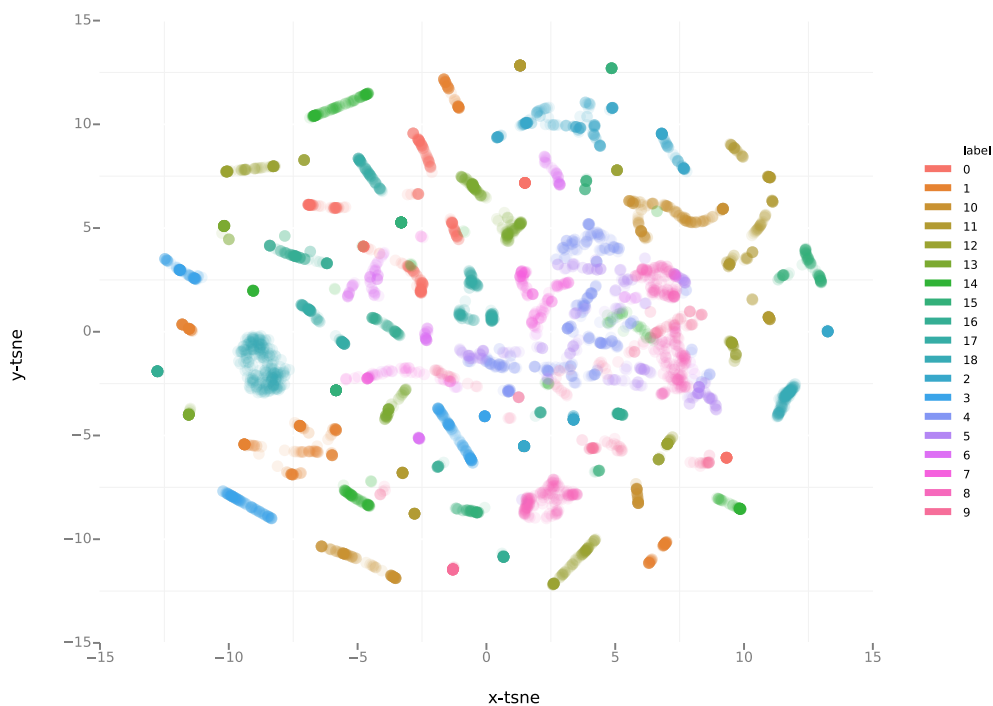
In another implementation, we composed is based on a widely used technique with minimal pre-processing and classifier ensemble for faster classification on a very high dimensional dataset such as multi-sensor EEG signal [36]. We attempt to dodge the curse of dimensionality that arises due to the large number of channel inputs through a series of assessments as described later in this section.

#### Visualizing High-Dimensional Data

A lack of efficacy and problems with representing high dimensional data is a well known issue, since a paper or screen is only a two-dimensional space. Perspective methods



can be used for a three-dimensional space, but visualizing for higher dimensional space is an unsolved challenge. Currently, there are two widely used approaches to support high-dimensional data visualization: (1) Parallel Coordinates: The loss of information is very low in this method, but perceiving information is extremely hard for viewers due to the level of clutter caused by the mass of overlapping lines, and (2) Dimensionality Reduction: As the name suggests, this is an approximate representation of original high dimensional data into any lower dimensional space that makes it very readable while preserving most of the information.



**Figure 5.3:** Dimensionality Reduction using t-Sne with 600 channels.

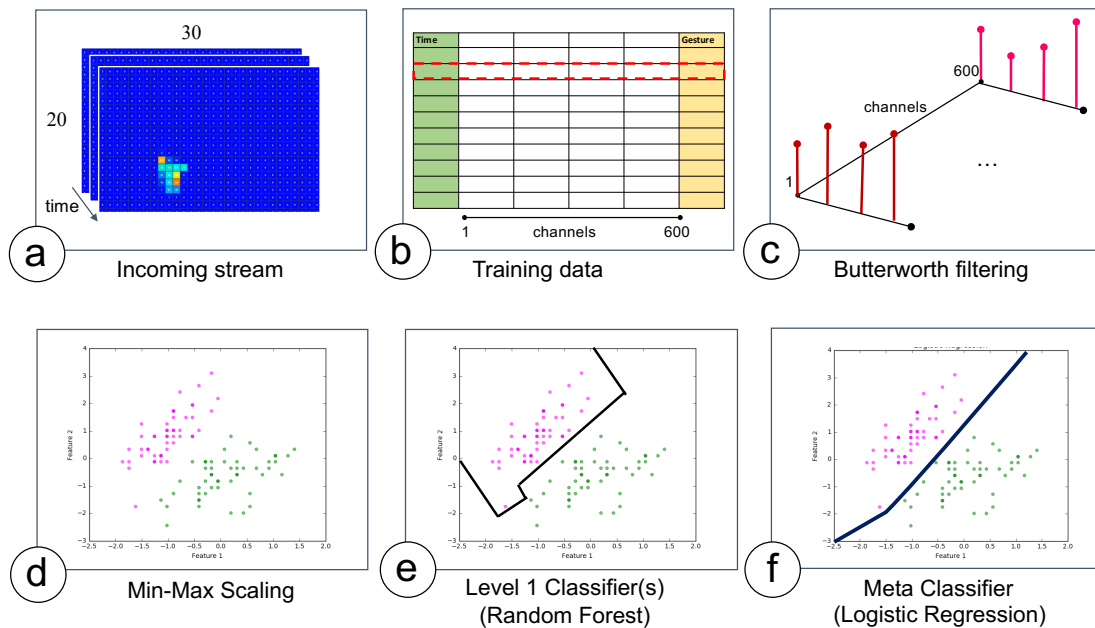
To describe the primary relationship amongst different gesture clusters, we adopted the dimensionality reduction technique using the raw channel data. We analysed results from several dimensionality reduction algorithms, including Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA), and the recent t-distributed Stochastic Neighbor Embedding (t-SNE) to draw clear inferences. In the end, we selected t-SNE as it clearly shows the relationship between different gestures, combined with clustering (Figure 5.3).

### Preprocessing and Feature Extraction

As almost all channels gets affected for some or the other gestures like, swipe activates more channels than a finger touch. Therefore, information from each channel is required, hence we treat each channel as an independent information source. Features are extracted from all the channels individually by applying a second-order Butterworth IIR low-pass filter, and a 101 sample length Hanning window that suppresses noise and smoothen the signal. Finally, we perform feature scaling by translating the obtained values between 0 and 1 in order to optimize the classification accuracy. These features are then used as an input to our model.

### Ensemble Learning

Given that our data is discretely clustered to some extent and we also do not apply any sophisticated filtering in order to effectively reduce the computational workload, we experimented with ensemble of classifiers with different combinations of classifiers for improving prediction performance. Although, ensemble learning have been empirically proven to improve accuracy in the field of Machine Learning with large class sets and noisy input [10], they are rarely applied in the field of HCI.



**Figure 5.4:** Stacking Cross-Validation algorithm.

Typically, there are three main ensemble methods: (1) Bagging: Training base classifiers on random instances of the overall training dataset, and then combine (voting or averaging) their individual predictions to form a final prediction, (2) Boosting: Creates a series of classifiers on the training set for which the mis-classification is reported by the

previous classifiers, and (3) Stacking: Also termed as Blending, combines the prediction from several models through a meta-classifier.

In order to use diverse set of base classifiers for a large number of classes, we selected Stacking technique. The standard stacking procedure is susceptible to overfitting as the same training set, which is used in first-level classifier is also an input for the second-level. Adding Cross-Validation (CV) with stacking can resolve this problem, since CV splits the dataset into N equal folds and uses N-1 for training, and calculates accuracy on the remaining one. These predictions are then used as an input for the latter-level classifier. Figure 5.4 illustrates the steps involved in stacking ensemble with CV.

We use a StackingCVClassifier implementation provided by the MLxtend (machine learning extensions) library [51] with random forest as first-level and logistic regression as a meta-classifier. We experimented with a number of classifiers on the basis of noise-tolerant behavior and operability with very high dimensional data, namely Logistic Regression (LR), Support Vector Machine (SVM), Linear Discriminant Analysis (LDA) and Random Forest (RF) (see Listing 6).

```

1  from sklearn.linear_model import LogisticRegression
2  from sklearn.ensemble import RandomForestClassifier
3  from mlxtend.classifier import StackingCVClassifier
4  #more classifiers can be added
5  clf1 = RandomForestClassifier(random_state=42)
6  clf2 = LogisticRegression()
7  stack = StackingCVClassifier(classifiers=[clf1],
8  meta_classifier=clf2, random_state=42)
9  #similar to standard scikit-learn classifier
10 stack.fit(X_train, y_train)
11 y_test = stack.predict(X_test)

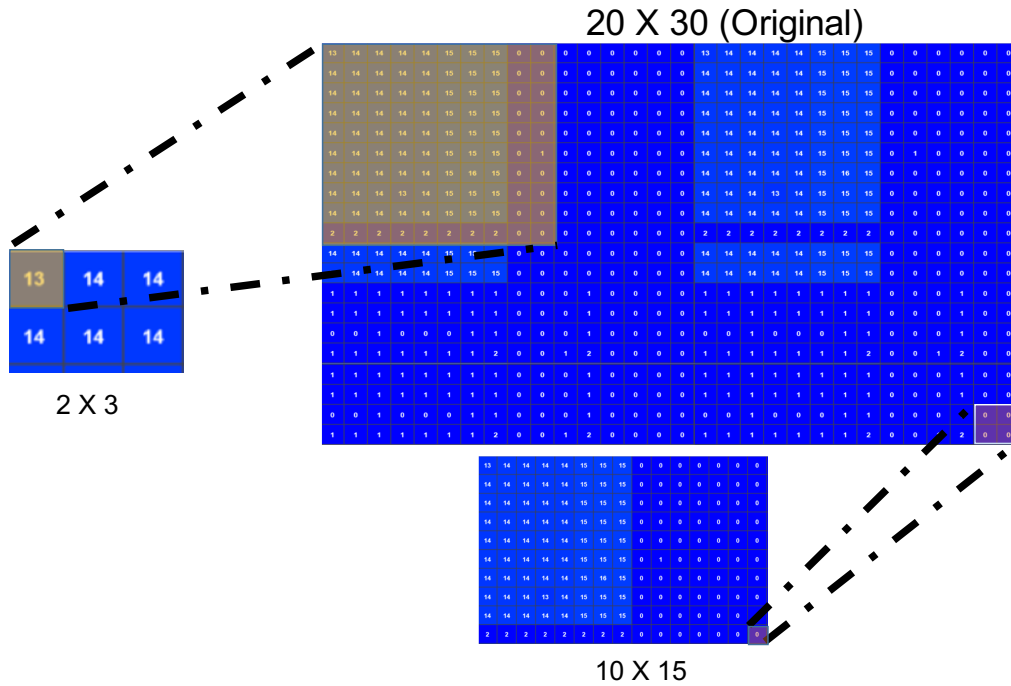
```

**Listing 6:** Python code for StackingCVClassifier.

The high resolution of the textile requires fast and easy methods to distinguish between the many classes of the gestures. To provide a high accuracy, all dimensions (2D position, pressure, time) were used for the recognition of all the gestures. Besides that, we were also interested in getting the *location*, the *direction* and the *pressure* information of a gesture as this supports to differentiate between even more gesture possibilities and shows how the input space can be used beneficially.

### Resolution Downscaling

Our objective here is to estimate a distribution over sensor values that can be used to compute the likelihood of a gesture. As mentioned earlier, the original dimensionality of sensor data can be seen as large matrices, which is used as a training data for our machine learning model. Although, reducing the size of big-data matrices can be performed through a variety of sophisticated statistical approaches, we used a simple matrix elements summation technique to minimize the matrix size (as shown in Figure 5.5). To evaluate the classification accuracy with the new scaling, we reused the



**Figure 5.5:** Resolution Downscaling.

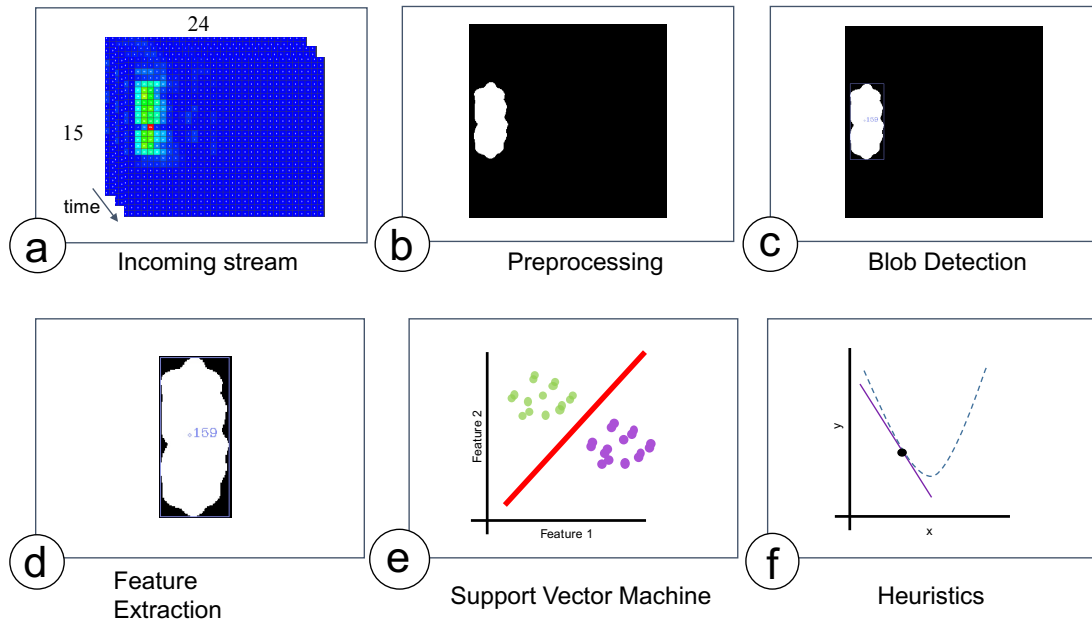
data and machine learning architecture from second model.

Although this strategy generates good accuracy, it requires laborious training for each gesture and also does not support space invariance, i.e. gesture trained at one location cannot be tested at any arbitrary location without training at that position. This model highlights an unseen constraint of machine learning in wearable technology, since building generic models is a difficult task due to the personalization factor (anatomical structure varies from one person to the other). Also, in our initial studies, we found asymmetry in the inference of gestures by individuals, we suspect that it is due to the influence of different mental models.

To overcome these challenges, we built a modular approach which allow *location*, *direction* or *pressure* variance or invariance whenever it is needed. This reduces the training trials as well as allows the user to use the same gestures to express different meanings, for example a slide gesture on the under arm can be used for the next song, while a slide on the upper arm play the next play list. Further, the locations variance allows the user to map most recent gestures on the lower arm while less recent gestures are placed on the upper arm [21]. Additional to the location variance, pressure can be used to gain or change the interpretation of the gesture. As an example, a low pressure slide be used for moving over the an interactive map while applying more pressure allow the user to move slower with more precession.

### 5.2.3 Model 3: Hybrid Approach

The realization from first two strategies underscores the key advantage of the third strategy - Our novel algorithm addresses the challenges of learning-based methods that need extensive training, as well as the extraction of hand-crafted features which require domain-specific knowledge, for class-label prediction. To alleviate these issues, we combine the results from the trained model with a heuristic-based approach that relies on a set of rules. Often used for applications, such as spam detection, filtering, and rule-based assessment, Heuristic-based models provide a unique capability to react to untrained classes. So, we can train a much smaller set of trained gestures and achieve a wide variety of different interactions. Furthermore, our method can detect the position of the gesture, the direction of the path for directional gestures, and the pressure level, which could be mapped for applications like controlling the scrolling speed.



**Figure 5.6:** SmartSleeve gesture detection pipeline. Receive raw sensor data (a) preprocess through foreground and background separation, as well as Gaussian filtering (b) blob detection (c) feature extraction from contours. (d) input features to classifier (e), and add the heuristic rules (f).

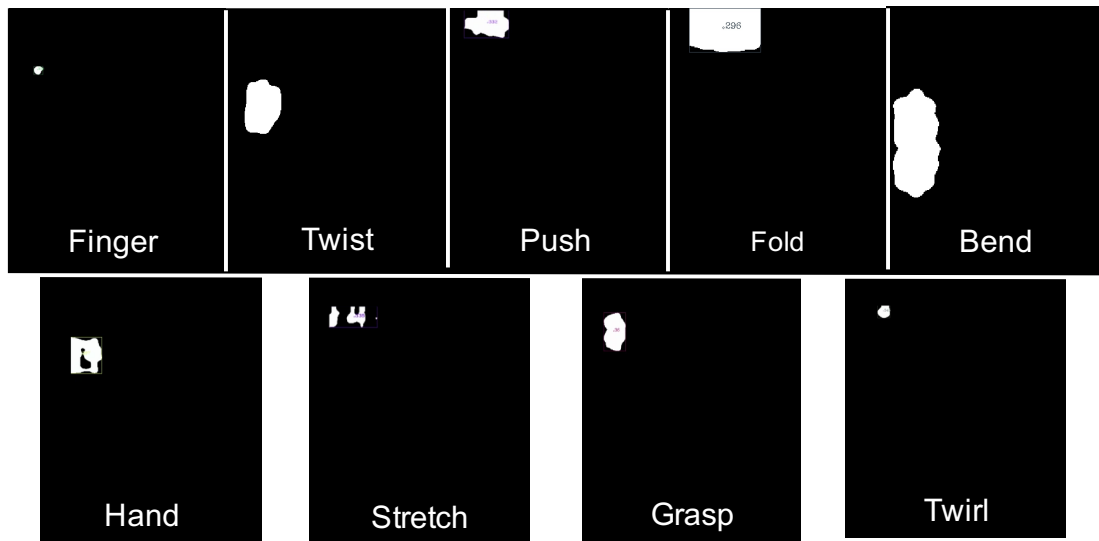
Figure 5.6 presents the entire SmartSleeve gesture detection pipeline. Our approach has some similarities with the method used in GravitySpace [3]. Similarly, we reduce a 3D problem to a 2D problem by constructing a force image using the raw sensor matrix data. Next, we construct a force image with a size of  $15 \times 24$  px with an overall framerate of 30 fps. Our algorithm is training a classifier on a per-frame-basis without any sliding window for temporal dynamics. Similar to the approach *Type-Hover-Swipe* [66], we have developed a simple filter, which averages the current sample with the previous ten samples to handle the *false* deformations on the *SmartSleeve*. Consequently, we can stabilize the natural tremor of hands and obtain temporal information. In addition, we

leverage our continuous gesture tracking mechanism for directional information. Particularly, we incorporate the results from learning-based algorithm and heuristic-based approach to reduce the training of gestures by half as well as decrease computationally expensive feature extraction.

### Preprocessing and Feature Extraction

Initially, we convert the raw sensor data to a grayscale force image. By applying a threshold we remove the noise. Further, we specify the foreground and background on an individual pixel to search for points of interest, which in turn results in the loss of the pressure information. However, this information is later recovered by calculating the average force from the raw sensor data, once the Region of Interest (RoI) is located. In the next step, we apply bilinear upscaling and a Gaussian filtering for smoothing the raw force image. The RoI is selected as a mask inside the bounding box using the blob detection model. We use the contour detection algorithm [13], which makes the gesture classification space invariant.

The removal of the pressure information from the force image yielded significant improvements in terms of classification accuracy during an informal pre-study with different users. Additionally, it helps us reducing the number of training trials, as the processed images for feature extraction appear similar even when the applied force changes up to a certain limit for a particular gesture during the training phase.



**Figure 5.7:** Training input frames after preprocessing.

As all the regions are highly discriminative (see Figure 5.7), we only compute a simple histogram and a set of properties of the contour's bounding box, including *height*, *width*, *area*, and *perimeter*, as features for the classification without any further processing. Since multiple contours can appear forming one gesture, we merge them into one result.

### Classifying Gestures Based on Image Analysis

In order to identify the gestures, we took a subset from Figure 3.1 based on their variances, namely *Finger*, *Hand*, *Bend*, *Twist*, *Push*, *Grasp*, *Stretch*, *Twirl*, and *Fold*. We experimented using the image features which we extracted above for these nine gestures as input to different classifiers and found that a Support Vector Machine (SVM) yields the most promising results, with parameters  $C = 1.0$  and  $kernel = RBF$ , optimized using a grid search with cross-validation implementation provided by the *scikit-learn* [44], mentioned in Listing 7. The model assigns probabilities for each type of trained gesture.

```

1  from sklearn.model_selection import GridSearchCV
2  from sklearn import svm
3  parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
4  clf = svm.SVC(probability=True)
5  classifier = GridSearchCV(clf, parameters)
6  prdiction = classifier.fit(X_train, y_train).predict(X_test)

```

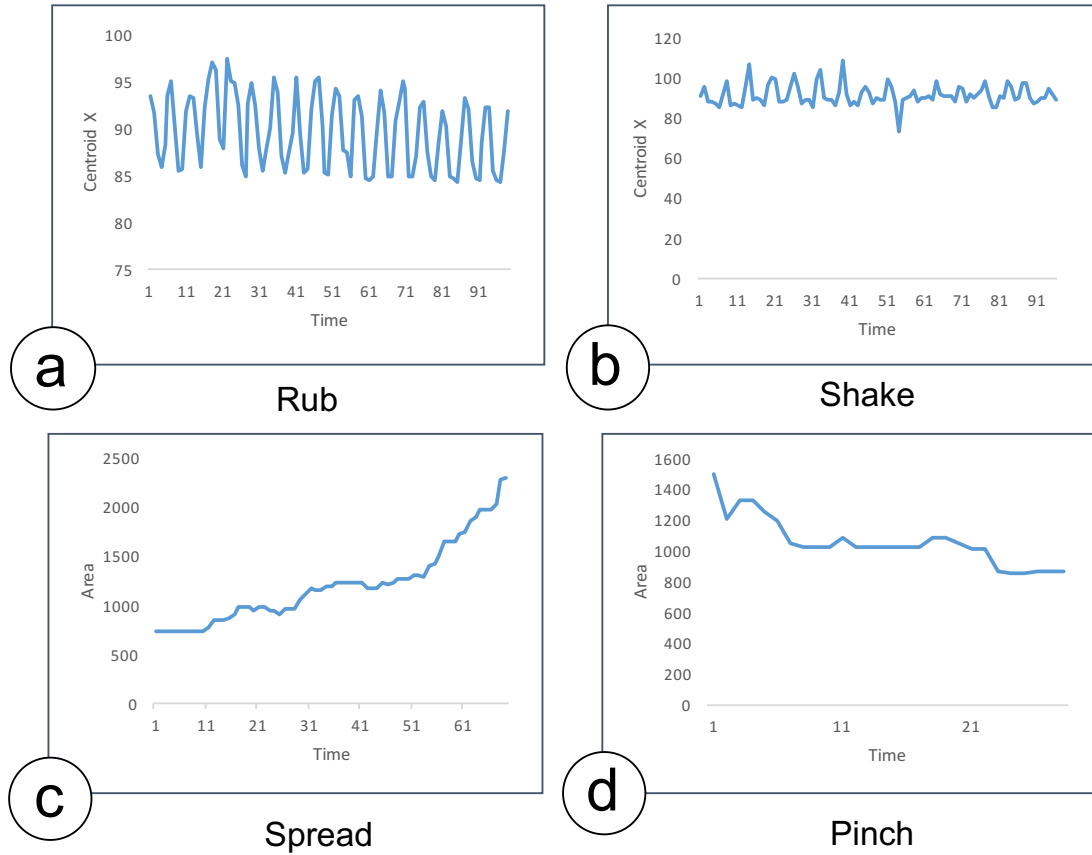
**Listing 7:** Python code for Tuning the hyper-parameters of a classifier using GridSearch.

### Detecting Untrained Gestures Using Heuristics

We extend our frame-by-frame learning-based algorithm for recognizing dynamic (un-trained) gestures by adding a set of rules based on the classifier generated probability distribution. This idea is inspired by the *Pose Recognition* technique used in GravitySpace [3]. In our implementation, we combine the highest probability with net force and properties of the blob (*position* and *size*) to produce more gestures (*finger swipe up, down, left, right, and rub; hand swipe up, down, left, right, and rub; spread; pinch; shake*). As mentioned earlier, we deduce the normalized force from the raw sensor data within a blob and our frame averaging helps us to track the blob within successive image frames.

Specifically, to identify the direction (*up, down, left, or right*) of the gesture, we simply store the consecutive blob's centroid coordinates in a buffer and compute the slope through each pair of points. Additionally, we implemented a consistency check algorithm to overcome the accidental deviations in slope values since a user might not be able to draw a straight line (e.g., left to right gesture) and the slope might have some unintended motions, which can be interpreted as an up- or down-gesture. We fix this problem by splitting the whole gesture into overlapping regions and further ignore small deviations.

Probability distributions of *grasp* and *hand* from the classifier help determine *shake* and *rub* respectively, the blob's centroid coordinates change significantly during *rub* and *shake*, cf. Figure 5.8, (a) and (b) compared to other gestures, we use the concept of first order derivative from calculus to compute this rate of change, since the obtained value is scalar, we make it absolute and take an average on a finite size to avoid false positives. Gestures including *spread* and *pinch* exploit the classification probability of



**Figure 5.8:** (a, b): Change in contour’s centroid during rub and shake, and (c,d): Change in area of the bounding box while performing spread and pinch over time.

*finger* combined with the linear change in area of contour (see Figure 5.8 (c) and (d), we measure the trend of increment or decrement with simple subtraction between pairs of consecutive area values. Afterwards, we assess the average to determine the action robustly wherein a positive value signifies expansion, and vice versa.

Our system supports 2.5D input based on 2D touch input and an additional pressure sensing level comparable with other systems [29, 53, 56]. Using the 2.5D depth map SmartSleeve supports touch gestures, such as touching or moving. Besides that, *SmartSleeve* detects gestures on base of the deformation of the fabric. When the user presses down, the sensor value changes. For example, when the users grabs, the fabric is pulled together and thus the pressure on specific points changes. SmartSleeve provides continuous pressure information as additional property to the gesture classification. This alleviate the issue of inconsistent pressure sensation in different contexts (also reducing training trials), and can be further used to detect the magnitude of stretch and other gestures.

Summarizing, our real-time recognition pipeline provides *continuous* gesture detection. In particular, if we attempt a swipe gesture, we have an additional component of



changing pressure along the line. This feature allows us an additional input modality and could be used for controlling speed in activities like *fast forward* in a video player. Our system also detects a few multi-touch gestures. Most typically, we recognize spreading for zooming into a map application or into a three-dimensional model. Therefore the position can be used to set the zooming position or the camera and the pressure sensing level indicates the speed. Additionally, as typically performed with the thumb and the index finger, this gesture can be also performed with the thumb and all other fingers, which can be used for a rough zooming.

### Saving and Restoring the Model

Once the model is trained, it is useful to save it on a disk for further predictions without training it again. We use the *joblib* module to save and load the model. This saves computation by making predictions on the new dataset without retraining every time. Also, the saved model can be easily transported to another device. Listing 8 represents how to persist and load a model.

```
1 from sklearn.externals import joblib
2 #saving
3 joblib.dump(classifier, 'model.pkl')
4 #restoring
5 clf_saved = joblib.load('model.pkl')
```

**Listing 8:** Python code for Model Persistence.

In addition to classifying a gesture, SmartSleeve detects three properties: The *Location (L)* where the gesture is performed on the sleeve, its *Direction (D)* and its *Pressure (P)* intensity. Note that not all gestures can use all properties: the *Bend* gesture, for example, can detect the pressure intensity, but its location is fixed at the user's elbow joint. Overall, these properties improve the quality of the gesture recognition, but they can also be used as design parameters. Using a property like *Direction* considerably expands the possible gesture set. To make use of these properties for certain gestures, we use our hybrid gesture detection approach, which takes advantage of the properties where appropriate.

As mentioned before, our hybrid algorithm can recognize all 22 gestures (7 *Deformation Gestures* + 2 *Surface Gestures* + 1 derived *Deformation Gesture* + 12 derived *Surface Gestures*) in real-time.

## Chapter 6

# Example Applications

These examples illustrate how 2D surface gestures, 2.5D deformation gestures, and the three continuous properties (location, direction, pressure) have the opportunity to greatly expand the opportunities for linking and mapping information while taking into account nuanced properties, such as, recency and importance [24, 72]. The proposed gestures can significantly increase the contextual appropriateness in various scenarios (as seen in figure 6.1).



**Figure 6.1:** SmartSleeve envisions seamless integration of interactivity in remote, social and private scenarios.

### 6.1 Surface Gestures: 2D Finger and Hand motion

Previous work has shown that users tend to transfer conventional multi-touch gestures to other modalities—especially for standard commands [32, 67, 72]. Therefore, we find it important to support a broad set of *Surface Gestures*, as depicted earlier in Chapter 3 (Figure 3.1). By making use of location, direction, and pressure properties, we are able to derive even more gestures, as shown later in Figure 7.4. In the case of the derivatives,

we distinguish between the following:

### Swipe

2D motion with the finger or hand on the sleeve affords relative or absolute positioning. The system can thus support traditional touch interactions, where surface interactions are mapped to, e.g., navigation, scrolling and panning. The ability to distinguish between finger and hand makes it possible to differentiate between coarse and fine control. In addition, spatial differentiation between input regions can extend the interaction space. For example, in a 3D modelling application, a movement across the upper arm could mean a rotation around the y-axis, while the same movement across the lower arm could be recognized as a rotation around the x-axis. Our eyes-free media player uses finger left/right swipes to skip forward/backward in a track, while a left/right swipes with the hand changes track. When our media player is used with visual feedback, finger motion can be used for cursor control, and swiped for menu option navigation.

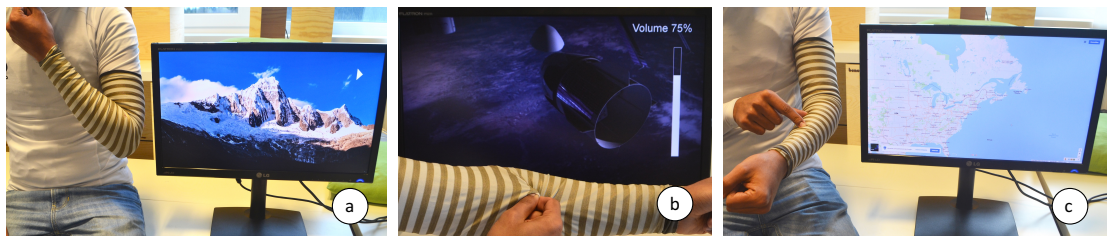
### Rub

1D back-and-forth motion with the finger or hand matches the metaphor of scratching something out with a pen. We thus found it attractive to map it to deletion. It could be used for deletion of an element, like dismissing a message or deleting a calendar entry [76]. In addition, the pressure intensity can be used to delete one or a whole set of items at once. In our media player, rubbing removes the current track from the playlist.

### Spread/Pinch

These gestures are widely used in ‘pinch-to-zoom’ interactions on multi-touch devices. They are derived from the *Finger* gesture and make use of location and direction (see Figure 7.4). While these gestures use multiple fingers, the algorithm is the same. This interaction is applicable to scalable interfaces with visual feedback, such as, for map navigation, and image manipulation.]

These commands except spreading and closing can be performed by one, multiple fingers or the full hand depending on the gesture. Single or multiple fingers can be used for high-level commands, while full hand would be used for low-level commands.



**Figure 6.2:** Gestures can be used to operate remote devices. In (a) and (b) the user starts media and changes the volume by performing bend and twist gestures, (c) spread can be used for controlling zoom level in map application.

## 6.2 2.5D Deformation Gestures

In addition to Surface Gestures, SmartSleeve enables a wide range of *deformable gestures*. The thin and elastic textile sensor material affords freeform manipulation and deformation, while our sensing technique detects gestures, state changes, and continuous manipulation.

### Twist

Pinching the textile and twisting it affords rotational control. The analogy to a physical knob makes it suitable for actions that map to clockwise or counterclockwise motion. In our media player, we use this gesture to increase/decrease the volume. The ability to sense location, also allows multiple virtual knobs along the textile—for example, to control an equalizer or left/right balance. Pressure might be used to control the rate (light touch would change the value more slowly). The physical constraints that prevent the gesture to be rotated infinitely match the physical affordances of control knobs that map to a value range. Our sensor currently does not support infinite rotation, which can be found in scroll wheels, or continuous rotary encoders.

### Push

Pushing the sleeve up can be treated as a state change, e.g., to hide information [41]. The compressed sleeve provides implicit visual and tactile feedback about the state. Our media player uses this state to toggle mute, or to hide the UI or media if used with visual feedback.

### Fold

Folding the sleeve is another way to change state. Here, we rely on the difference in *operation* to distinguish it from Push. While the end result may look similar, this operation requires careful effort to perform. With our media player, we map this operation to entering recording mode.

### Twirl

Twirling the textile around the finger requires intentional coordination. It uses the metaphor of the "reminder knot" around a finger. We use it to assign importance to the current item in the interface. The media player lets users rate a track by assigning a "star" or "like" with the gesture. When used with an audio book, podcast or radio show, it sets a bookmark. One could also imagine saving the currently playing voice mail message, or using it to record a voice memo. Location for the Twirl can be used to later enable retrieval with random access.

### Grasp

Grasping consists of the user grabbing the textile and pulling it together into the fist. We use it as a metaphor for retrieval. This, for example, allows us to complement Twirl

with a mechanism for activating a saved item. The location can be mapped to specify which saved item to retrieve.

### Shake

Shake is a derived gesture from *Grasping* (cf. Figure 7.4). The metaphor is based on grabbing a container with objects and shaking it. We map it to shuffling the tracks in our media player. Another considered mapping would be to clear the list or to close the application [32].

### Stretch

Stretching consists of pulling on the textile at a specific location. It affords elastic input as the textile retracts when released. We use the metaphor of turntable control, where stretching controls playback speed in our media player. Stretching it towards the user increases the speed, while pulling it away decreases the speed.

### Bend

Bending of the elbow is an example of the implicit sensing that is possible with our technique. As this motion is part of the user's natural movement, we would need to use a disambiguating mechanism, e.g., pressure or combination with another gesture, to activate it if used as an explicit command. Another opportunity is to use it as implicit input. For our media player, we have explored using the bending that occurs from arm swinging while running as a way to detect the appropriate tempo for the music playlist.

## Chapter 7

# Evaluation

Two empirical studies were conducted to evaluate our hybrid gesture detection algorithm. In the first experiment, we evaluated the learning based algorithm and we were primarily interested in finding out if the trained gesture set would also be position-invariant. On the other side, in the second experiment, we were focusing on the evaluation of the heuristic approach, where we wanted to find out if the heuristics we implemented will help to enrich the set of gestures while training only a subset of gestures.

### 7.1 Participants

Six unpaid volunteers (4 female), 23–36 years old ( $\bar{x} = 30$ ,  $\sigma = 3.8$ ), all right-handed were recruited from the local university. All participants used 2D touch interfaces on a daily base, but none of them had experiences with smart clothing interfaces.

### 7.2 Apparatus

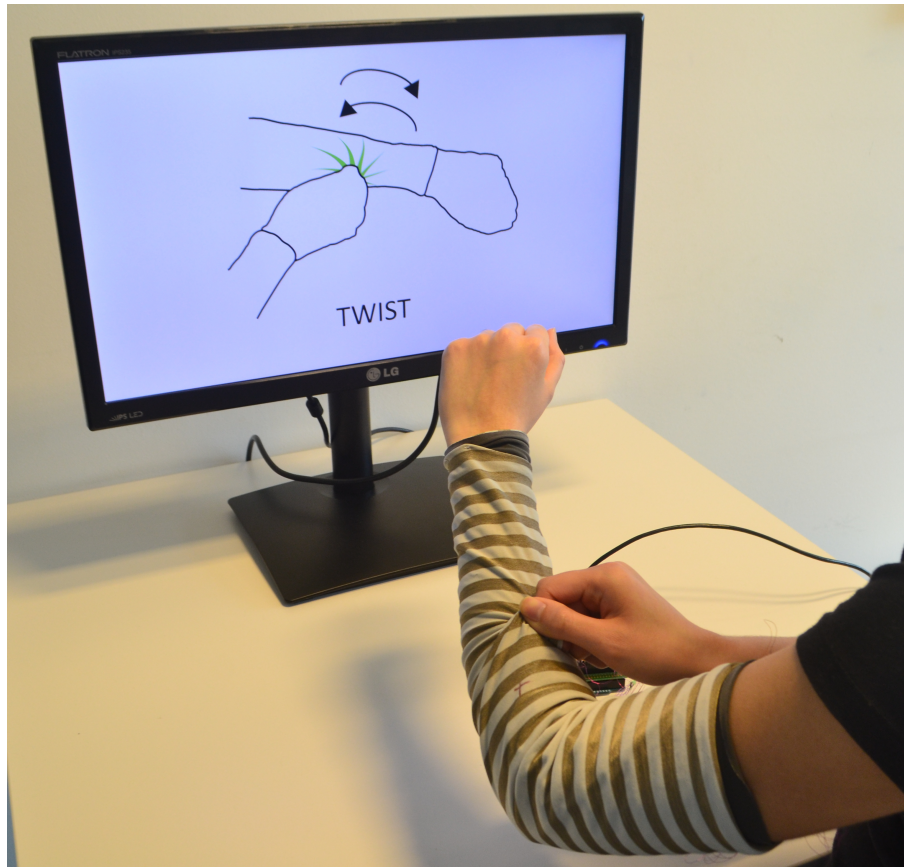
The study was conducted in a quiet room, where the participants were wearing the SmartSleeve clothing as depicted in Figure 7.1. The instructions were displayed on LG 24" 1920 × 1200 pixel IPS LCD screen.

### 7.3 Experiment 1: Position-Invariant Gesture Recognition

In the first experiment, we performed a gesture recognition experiment, where we wanted to find out if our approach also provides good accuracy results even if the gestures have to be performed on different locations of the textile.

#### 7.3.1 Design

At the beginning of the experiment, the nine different gesture types (i.e., *Finger*, *Hand*, *Twist*, *Bend*, *Stretch*, *Fold*, *Push*, *Grasp*, *Twirl*) were demonstrated to participants for clarity. Thereafter, participants were instructed to perform four trials of each gesture type in randomized order to train the gesture recognition engine.



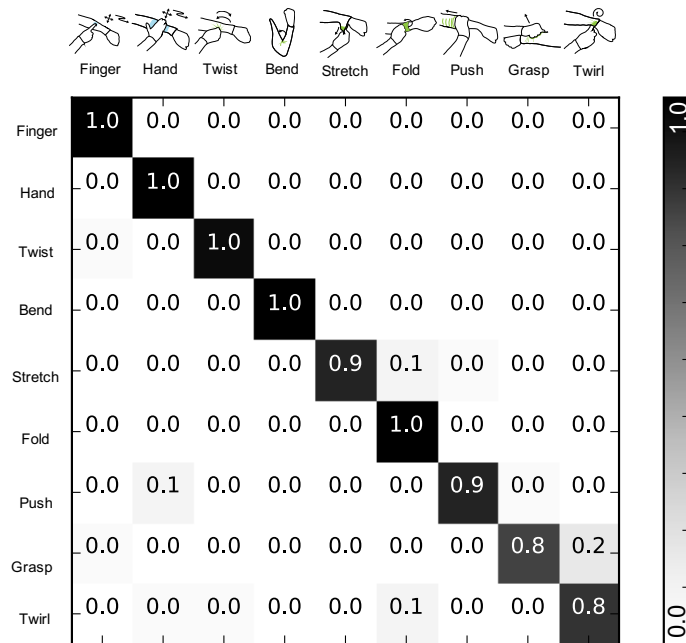
**Figure 7.1:** Apparatus for both experiments. All the gestures have been displayed on the screen and the participants performed the gestures using the SmartSleeve hardware accordingly.

All these four trials had to be performed on the same location, which has been marked accordingly with a red-sewn thread. The training phase took approx. 15 minutes. Next, participants were asked again to perform eight trials of each gesture type on the (a) same location (red-sewn thread position) as well as on (b) an arbitrary. The order of the gesture type was randomized and presented accordingly on the on-screen prompts (see Figure 7.1). The on-screen prompt further showed whether the gesture was recognized correctly or wrongly. The testing phase took about 35 minutes per participant. Collected measurements included error rate.

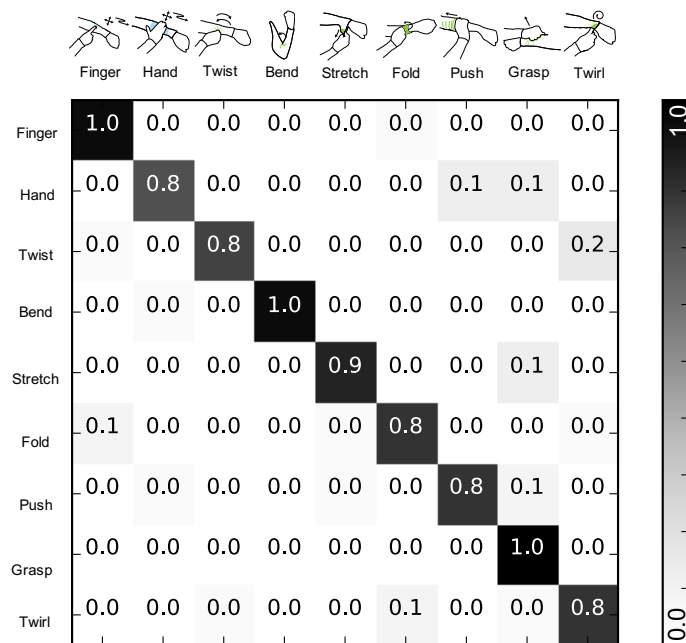
### 7.3.2 Results

Our approach reached an average accuracy of 92.0 % when the system is trained and tested by the same participant on the same location, cf. Figure 7.2. Similar results have been achieved when the system is trained and tested by the same participant on different location with an average of 86.9 %, cf. Figure 7.3.

In more detail, the finger gesture achieved an average recognition rate of 96.4%



**Figure 7.2:** The standard confusion matrix for the SmartSleeve hardware on the same training and testing location.



**Figure 7.3:** The standard confusion matrix for the SmartSleeve hardware using an arbitrary location on the arm chosen by the participants.

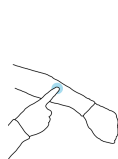
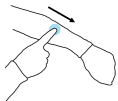
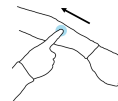
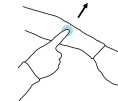
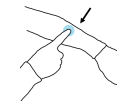


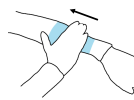
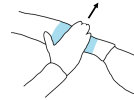
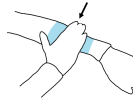




( $\sigma = .05$ ), 100% ( $\sigma = 0.0$ ) for the bend gestures, and 83.6% ( $\sigma = .167$ ) for the twirl gestures. Using different locations, an average recognition rate of 92.6% ( $\sigma = .121$ ) for the finger gestures, 98.0 % ( $\sigma = .05$ ) for the bend gestures, and 80.0% ( $\sigma = .244$ ) for the twirl gestures was achieved.

A repeated measures ANOVA was carried out and revealed a significant effect for the location ( $F_{1,35} = 6.019; p < .05$ ) as well as gestures ( $F_{8,35} = 5.865; p < .001$ ). Furthermore, a post-hoc analysis did not show any significant effects.

## 7.4 Experiment 2: Testing the Heuristics Approach

In the second experiment, we evaluated the performance of the heuristics of the gestures using the parameters *pressure*, *location* and *direction*. We therefore wanted to find out if we are able to recognize more complex gestures, which are based on simple ones.

Trained Gesture	Derived Gesture			
 <p data-bbox="416 1084 491 1115"><b>Finger</b></p>	 <p data-bbox="576 1010 705 1039"><b>Swipe Right</b></p>	 <p data-bbox="761 1010 890 1039"><b>Swipe Left</b></p>	 <p data-bbox="930 1010 1059 1039"><b>Swipe Up</b></p>	 <p data-bbox="1099 1010 1228 1039"><b>Swipe Down</b></p>
 <p data-bbox="424 1435 483 1467"><b>Hand</b></p>	 <p data-bbox="576 1364 705 1393"><b>Swipe Right</b></p>	 <p data-bbox="761 1364 890 1393"><b>Swipe Left</b></p>	 <p data-bbox="930 1364 1059 1393"><b>Swipe Up</b></p>	 <p data-bbox="1099 1364 1228 1393"><b>Swipe Down</b></p>
 <p data-bbox="416 1704 491 1736"><b>Grasp</b></p>	 <p data-bbox="616 1704 681 1736"><b>Shake</b></p>			

**Figure 7.4:** 3 trained classes are augmented to derive 13 new gestures using our heuristic approach.

### 7.4.1 Design

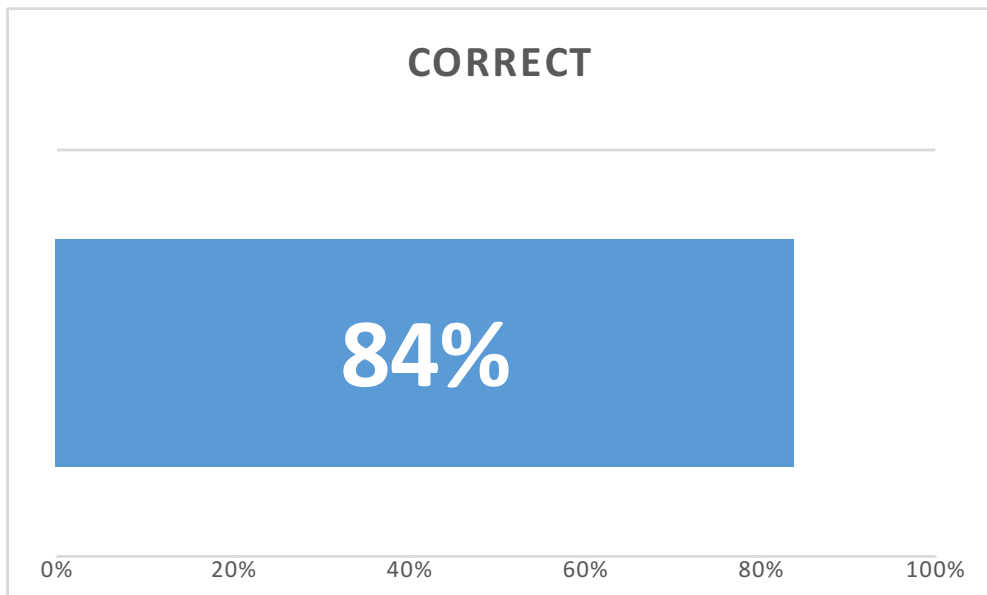
We evaluated the recognition implementation with the same participants. As all the used gestures were based on the gestures described before, no training data was required for this evaluation. Again, we showed each new gesture on a screen and the participant performed it accordingly.

For the second experiment, we chose a subset of the gesture classes (i.e., *Finger*, *Hand*, and *Grasp*), as all of them are making use of all the used parameters (*pressure*, *location* and *direction*), cf. Figure 7.4. Every participant performed each gesture 5 times in total, resulting in an overall of 80 trials per participant.

For the gesture class *Finger*, for instance, participants had to perform the gestures *Swipe right*, *swipe left*, *swipe up*, *swipe down*, *tab*, *rub*, *spead*, and *pinch* with the index finger on an arbitrary location of the SmartSleeve. The same gestures were performed using the *Hand* gesture class. Finally, participants also *grasped* and *shaked* the sleeve. All the gestures were counterbalanced to avoid training effects.

### 7.4.2 Results

Figure 7.5 shows robust results with 84% ( $\sigma = 0.11$ ) of all gestures that were correctly identified. In more detail, the simple gesture detection (*Swipe right*, *swipe left*, *swipe up*, *swipe down*, *tab*, *rub*, *spead*) using the finger achieved an average recognition rate of 83% ( $\sigma = .06$ ), while we achieved 97% for the *pinch* gesture. Only very complex deformation gestures, like the shake achieved an average recognition rate of 74%.



**Figure 7.5:** Results of 13 gestures detected by heuristic approach.

## 7.5 Technical Evaluation

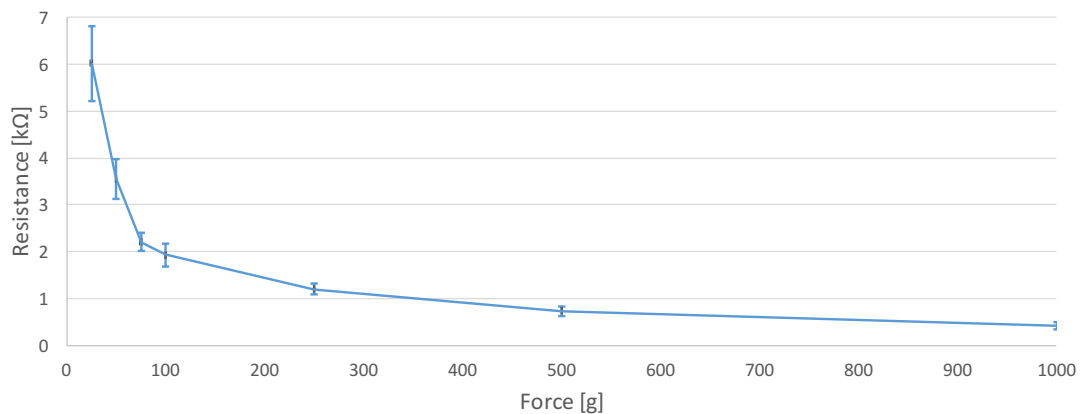
To evaluate the pressure sensing behaviour of the sensor we conducted a technical study, where we applied mechanical stress onto the surface of the SmartSleeve sensor.

### 7.5.1 Apparatus

The sensor got stationary mounted onto a flat deformable Styrofoam surface. A hemispherical thrust plate with a 4 mm diameter applied mechanical stress onto the surface of the sensor. The resistance change of the sensor was measured 10 times with a force of 25 g, 50 g, 75 g, 100 g, 250 g, 500 g, 1000 g.

### 7.5.2 Results

As seen in Figure 7.6 the sensor shows promising sensing behaviors from 50 g to 500 g. Below 50 g the sensor demonstrates high resistance changes as expected, however according to the loose stacking of the sensor the standard deviation is high. Beyond 500 g the resistance of the sensor shows slight changes, therefore disregard this area in SmartSleeve as well.



**Figure 7.6:** Pressure Sensing under different stress levels.

## Chapter 8

# Discussion and Limitations

The SmartSleeve was trained and tested for a period of four months in total. During that time, the sensor was used approximately 120 times by several persons who provided early feedback.

### 8.1 Durability

We observed that the sensor signal did not change significantly even under different pressure conditions. Regarding the connections, we implemented two versions. The first version of our sensor had a rough rigid connection using flat ribbon cables (2.54 mm pitch) connecting the sensor PCB board with the textile. The connections of this prototype broke relatively easily, as the cable were very stiff and heavy and the soldering spots were comparably small. Further, having this rigid soldered connection directly on the sleeve leads to breaks, while performing a highly deformable gesture activity. The second version of SmartSleeve with the new wire cables (as introduced in this paper) was then tested for approximately three months. During that time, the sleeve was used more than 100 times by different participants. Only three smaller issues had to be fixed. Nevertheless, we think that this is the most challenging point that has to be further addressed.

### 8.2 Pressure Input

As the results of the evaluation show, SmartSleeve accurately recognizes 2D gestures (*surface gestures*) as well as 2.5D deformation-based gestures (*deformation gestures*) that are performed on the textile. Even though our algorithm takes advantage of pressure to detect 2.5D gestures, we have not evaluated the pressure itself for *surface gestures*. The high pressure resolution of the sensor could be used for enhancing surface gestures. Yet, as noted by Rendl et al. [52], pressure is a very subjective property and its perception differs from person to person. Therefore, we did not formally evaluate this aspect in this paper.

### 8.3 Three-layer Approach

While the three-layer approach currently is the only possible solution for a tactile sensor that measures signals via a resistive approach, it was problematic in a few instances when the user grasped only the top layer for performing a gesture. Moreover, a three-layer sandwich is thicker, decreases the comfort of use and needs more implementation effort. Therefore we strongly push for a one-layer solution which we are currently developing.

### 8.4 Use Inside Clothing

Early tests have shown that SmartSleeve can also be used as an inner layer in clothing, protected by an additional layer on top. Therefore it is also possible to wear the sensor under jackets, pullovers or other clothing. The stiffness and deformability of the outer textile defines which gestures are still possible to perform with SmartSleeve. In addition to use on or inside clothing, SmartSleeve can act as a textile input sensor for interaction with a wide range of interactive objects and devices.

### 8.5 Tailored Clothing

SmartSleeve is tailored for a person having a wrist perimeter of approx. 16 cm, an elbow perimeter of 26 cm and an upper arm perimeter of approx. 26 cm. Although all three layers are bidirectionally stretchable, observations with other participants have shown that the sleeve itself has to fit tightly, as it has to follow the rotation of the arm. If only the upper arm and elbow are fitting well, but the sleeve itself is loose on the lower arm, people can rotate the underarm inside the sleeve, which could lead to reduced accuracy. Generally, as SmartSleeve is a personal wearable, it should be tailored for a person to enable rich input on clothes.














### 8.6 Personalized Device

As the user tests with different users have shown, people interpret and perform gestures differently. SmartSleeve enables a rich set of gestures, but it also requires consciousness of the person performing the gesture. Some gestures (e.g., Push and Fold) have to be trained and defined precisely, because they can interfere with other gestures, which is not supported by the algorithm so far. In this paper, we defined the gesture class *Bend* as discrete gesture modality, but it can also detect several levels of bending, which can further be combined with other gestures. As an example the sleeve could detect bending with 90° angle of the elbow with a *Twist* gesture. Generally, we have seen that a large set of gestures also requires a precise definition of gesture and conciseness performing the gesture.

## Chapter 9

# Conclusions and Future Work

We introduced *SmartSleeve*, a flexible textile sensor that senses both *surface gestures* and *deformation gestures* in real-time. Previous work has focused on conceptualizing new gestures [32, 67, 76], and on implementing one or a few gestures in a working system [19, 30, 41, 61]. SmartSleeve, however, provides a unified sensing framework which allows us to detect all of them within a single pipeline, as shown in Table 9.1. Furthermore, most of the gestures can be done at any location on the sleeve, in contrast to previous work, which restricts gestures to smaller, dedicated, instrumented areas.

Related work	Input space	Surface Gestures					Deformation Gestures							
		Wrist to upper arm					Forearm to upper arm				Wrist			
														
Surface Gestures [76]	Tabletop	o	o	o	o	o								
On-Body Interaction [22]	Skin						x							
More than Touch [72]	Skin	o		o				o	o					
PrintSense [18]	Film	x		x						x				x
Flexy [68]	Film													x
iSkin [73]	Film	x	x	x	x	x								
Pinstripe [30]	Textile								x					
GestureSleeve [61]	Textile	x	x											
Deformable displays [32]	Textile							o	o	o		o		o
Elastic displays [67]	Textile							o	o					
Grabbing at an angle [19]	Textile								x					
AugmentedForearm [41]	Textile												x	
SmartSleeve	Textile	x	x	x	x	x	x	x	x	x	x	x	x	x

**Table 9.1:** The SmartSleeve gesture set compared with previous work (o = conceptual, x = functional).

We provided a detailed description of our hybrid gesture detection pipeline that uses learning-based algorithms and heuristics to enable real-time gesture detection and tracking. Its modular architecture, combined with a large sensor size, a high spatial resolution and a high pressure resolution, allowed us to derive new gestures through the combination with continuous properties like *pressure*, *location*, and *direction*.

Finally, we reported on the promising results from our evaluations which demonstrated real-time classification of 9 gestures with 89.5% accuracy.

In future work, we plan improvements to our SmartSleeve hardware, as the current proof-of-concept implementation relies on a wired PC connection for data transmission and power supply. Given the current hardware's dimensions ( $102 \times 53 \times 25mm$ ), miniaturization of the electronics would allow us to embed it in the textile. Therefore, we are developing a version, which is completely mobile with wireless connectivity (e.g., Bluetooth/WiFi). Additionally, we are working on a single-layer textile sensor implementation and implementing the algorithm on a hardware level.

We also want to explore how SmartSleeve performs in everyday scenarios and how environmental influences, like humidity, affect the sensor. Our initial experiments show that the sensor withstands machine washing at low temperature and slow spin, however, more formal evaluations should be performed to assess the durability of the sensor. Another goal is to replace the three layer approach, with a novel single layer textile, where the yarn itself is pressure sensitive.

Appendix A

## Study Questionnaire



**Welcome!**

First of all, thanks for your interest in our study. Before we can start, we want to give you a brief introduction about the goal of the study and what role you play. The duration of our study is one hour.

**Goal of the study**

---

As you might know, the object of our study is to get a deeper understanding of our interactive textile sleeve.

At this point, we want to emphasize that we do not test YOU, we test the performance of the device.

For the evaluation, it would be very helpful to record your training and test data as well as to archive the questionnaire. For sure, we will process the data anonymously. However, we need your consent, which we undertake in return to use the material only for evaluation and internal presentation purposes. We will draw attention that summary information and / or single citations could be presented in an upcoming publication.

In the unlikely case that something goes wrong in this study we advise you to contact us.

We wish you a lot of fun and we would like to thank you again for your participation!

**Consent Form**

Please read the following lines carefully.

In order to allow a better evaluation of the data obtained, we will record your gesture trials and the questionnaire. In turn, we commit ourselves to an anonymous recording and to use the material only for evaluation purposes. Furthermore, we would like to ask to preserve silence about the exact conduct of the study so as not to influence other potential participants.

So as to be able to perform the diary study, you will get a textile sleeve from us. Please take care of the sleeve.

At this point we would like to inform you about your rights during the investigation:

- You can abort the study at ANY TIME without negative consequences
  - In the unlikely case that something goes wrong in this study do not hesitate to contact us. However, please understand that we can respond to device-specific questions only after the test to prevent corruption of the data.
- 

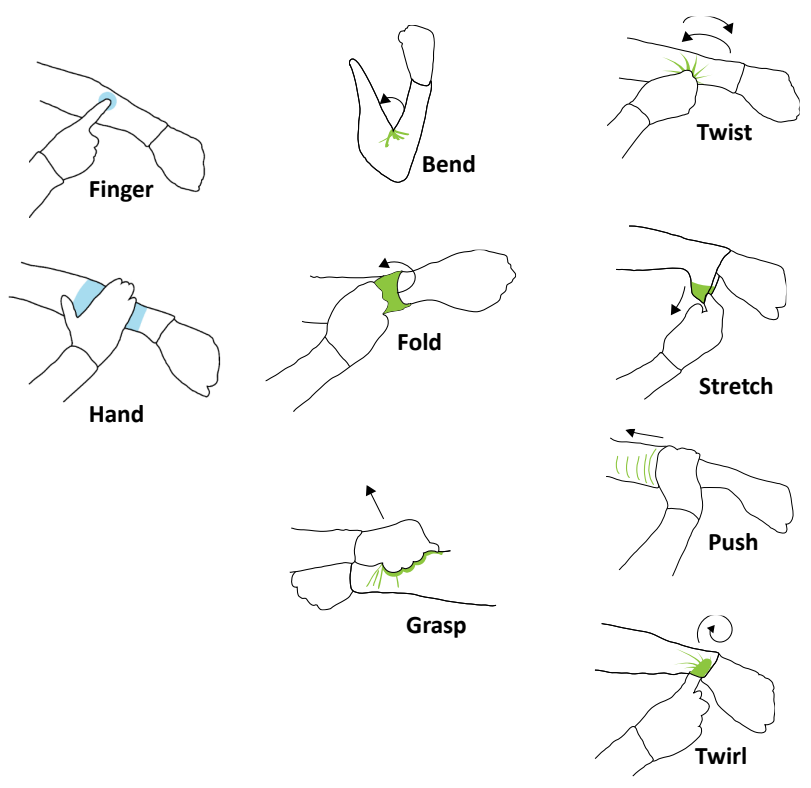
I have read and understood all of the above. I agree with all the points.

Name \_\_\_\_\_  
Signature \_\_\_\_\_  
Date \_\_\_\_\_

The study team undertakes, with its signature, to anonymize all data of this investigation and to use it only for evaluation and internal presentation purposes:

Name \_\_\_\_\_  
Signature of the study leader \_\_\_\_\_  
Date \_\_\_\_\_

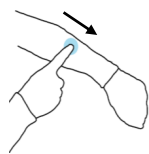
# Study 1



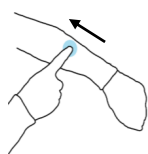




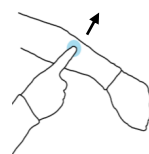
## Study 2



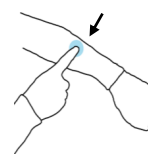
**Swipe Right**



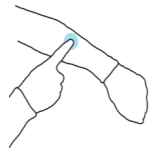
**Swipe Left**



**Swipe Up**



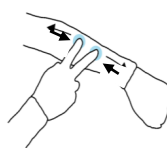
**Swipe Down**



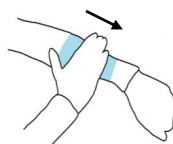
**Rub**



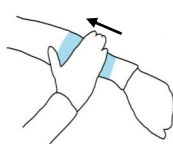
**Spread**



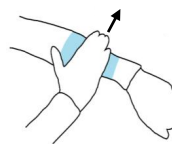
**Pinch**



**Swipe Right**



**Swipe Left**



**Swipe Up**



**Swipe Down**



**Rub**



## Appendix B

# DVD Content

**Format:** DVD-ROM, 4.7 GB

### B.1 PDF Data

**AS:** /

Sharma\_Adwait\_2017.pdf Master's thesis as PDF file.

### B.2 Video

**AS:** /

SmartSleeve-1080p.mp4 SmartSleeve Video



# References

## Literature

- [1] Daniel Lee Ashbrook. “Enabling mobile microinteractions”. Dissertation. Georgia Institute of Technology, 2010. URL: <http://smartech.gatech.edu/handle/1853/33986> (cit. on p. 7).
- [2] Patrick Baudisch and Gerry Chu. “Back-of-device interaction allows creating very small touch devices”. In: *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*. ACM Press, 2009, p. 1923. URL: <http://doi.acm.org/10.1145/1518701.1518995> (cit. on p. 7).
- [3] Alan Branzel et al. “GravitySpace”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, 2013, p. 725. URL: <http://doi.acm.org/10.1145/2470654.2470757> (cit. on pp. 8, 29, 31).
- [4] Leah Buechley and Michael Eisenberg. “Fabric PCBs, electronic sequins, and socket buttons: techniques for e-textile craft”. *Personal and Ubiquitous Computing* 13.2 (Feb. 2009), p. 133. URL: <http://dx.doi.org/10.1007/s00779-007-0181-0> (cit. on p. 17).
- [5] Leah. Buechley et al. *Textile Messages: Dispatches From the World of E-Textiles and Education*. 2nd ed. Peter Lang, 2013, p. 246. URL: <http://dx.doi.org/10.3726/978-1-4539-0941-6> (cit. on p. 2).
- [6] William Buxton. “Lexical and pragmatic considerations of input structures”. *Computer Graphics* 17.1 (Jan. 1983), p. 31. URL: <http://doi.acm.org/10.1145/988584.988586> (cit. on p. 9).
- [7] Lina M Castano and Alison B Flatau. “Smart fabric sensors and e-textile technologies: a review”. *Smart Materials and Structures* 23.5 (May 2014). URL: <https://doi.org/10.1088/0964-1726/23/5/053001> (cit. on p. 2).
- [8] Jared Cechanowicz, Pourang Irani, and Sriram Subramanian. “Augmenting the mouse with pressure sensitive input”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*. ACM Press, 2007, p. 1385. URL: <http://doi.acm.org/10.1145/1240624.1240835> (cit. on p. 6).
- [9] Tommaso D’Alessio. “Measurement errors in the scanning of piezoresistive sensors arrays”. *Sensors and Actuators A: Physical* 72.1 (Jan. 1999), p. 71. URL: [https://doi.org/10.1016/S0924-4247\(98\)00204-0](https://doi.org/10.1016/S0924-4247(98)00204-0) (cit. on p. 19).

- [10] Thomas G Dietterich. “Ensemble Methods in Machine Learning”. In: *International workshop on multiple classifier systems*. Springer, 2000, p. 1. URL: <http://dl.acm.org/citation.cfm?id=648054.743935> (cit. on p. 26).
- [11] Lucy E. Dunne et al. “Multi-layer e-textile circuits”. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*. ACM Press, 2012, p. 649. URL: <http://dl.acm.org/citation.cfm?doid=2370216.2370348> (cit. on p. 18).
- [12] Sean Follmer et al. “deForm”. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, 2011, p. 527. URL: <http://dl.acm.org/citation.cfm?doid=2047196.2047265> (cit. on p. 6).
- [13] Fu Chang and Chun-Jen Chen. “A component-labeling algorithm using contour tracing technique”. In: *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings*. Vol. 1. IEEE Comput. Soc, 2003, p. 741. URL: <https://doi.org/10.1109/ICDAR.2003.1227760> (cit. on p. 30).
- [14] Markus Funk et al. “Using a touch-sensitive wristband for text entry on smart watches”. In: *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems - CHI EA '14*. ACM Press, 2014, p. 2305. URL: <http://dl.acm.org/citation.cfm?doid=2559206.2581143> (cit. on pp. 5, 7, 10).
- [15] F. Gemperle et al. “Design for wearability”. In: *Proceedings of the 2nd IEEE International Symposium on Wearable Computers - ISWC '98*. IEEE Computer Society, 1998, p. 116. URL: <http://dl.acm.org/citation.cfm?id=857199.857998> (cit. on p. 4).
- [16] Guido Gioberto et al. “Detecting Bends and Fabric Folds Using Stitched Sensors”. In: *Proceedings of the 17th annual international symposium on International symposium on wearable computers - ISWC '13*. ACM Press, 2013, p. 53. URL: <http://doi.acm.org/10.1145/2493988.2494355> (cit. on p. 5).
- [17] Antonio Gomes, Andrea Nesbitt, and Roel Vertegaal. “MorePhone”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, Apr. 2013, p. 583. URL: <http://dl.acm.org/citation.cfm?doid=2470654.2470737> (cit. on p. 6).
- [18] Nan-Wei Gong et al. “PrintSense: A Versatile Sensing Technique to Support Multimodal Flexible Surface Interaction”. In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, 2014, p. 1407. URL: <http://doi.acm.org/10.1145/2556288.2557173> (cit. on p. 46).
- [19] Nur Al-huda Hamdan et al. “Grabbing at an Angle: Menu Selection for Fabric Interfaces”. In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers - ISWC '16*. ACM Press, 2016, p. 1. URL: <http://doi.acm.org/10.1145/2971763.2971786> (cit. on pp. 5, 10, 46).
- [20] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. “OmniTouch: Wearable Multitouch Interaction Everywhere”. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, 2011, p. 441. URL: <http://doi.acm.org/10.1145/2047196.2047255> (cit. on p. 7).

- [21] Chris Harrison and Haakon Faste. “Implications of Location and Touch for On-body Projected Interfaces”. In: *Proceedings of the 2014 conference on Designing interactive systems - DIS '14*. ACM Press, 2014, p. 543. URL: <http://doi.acm.org/10.1145/2598510.2598587> (cit. on pp. 4, 12, 28).
- [22] Chris Harrison, Shilpa Ramamurthy, and Scott E. Hudson. “On-body Interaction: Armed and Dangerous”. In: *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction - TEI '12*. ACM Press, 2012, p. 69. URL: <http://doi.acm.org/10.1145/2148131.2148148> (cit. on pp. 5, 46).
- [23] Chris Harrison, Desney Tan, and Dan Morris. “Skinput: Appropriating the Body As an Input Surface”. In: *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, 2010, p. 453. URL: <http://doi.acm.org/10.1145/1753326.1753394> (cit. on p. 7).
- [24] Chris Harrison et al. “Where to Locate Wearable Displays?: Reaction Time Performance of Visual Alerts from Tip to Toe”. In: *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*. ACM Press, 2009, p. 941. URL: <http://doi.acm.org/10.1145/1518701.1518845> (cit. on pp. 10, 34).
- [25] Florian Heller et al. “FabriTouch: Exploring Flexible Touch Input on Textiles”. In: *Proceedings of the 2014 ACM International Symposium on Wearable Computers - ISWC '14*. ACM Press, 2014, p. 59. URL: <http://doi.acm.org/10.1145/2634317.2634345> (cit. on p. 5).
- [26] Christopher F Herot and Guy Weinzapfel. “One-point Touch Input of Vector Information for Computer Displays”. *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques -SIGGRAPH '78* 12.3 (Aug. 1978), p. 210. URL: <http://doi.acm.org/10.1145/965139.807392> (cit. on p. 6).
- [27] David Holman and Roel Vertegaal. “Organic User Interfaces: Designing Computers in Any Way, Shape, or Form”. *Communications of the ACM* 51.6 (June 2008), p. 48. URL: <http://doi.acm.org/10.1145/1349026.1349037> (cit. on p. 6).
- [28] Christian Holz et al. “Implanted User Interfaces”. In: *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. ACM Press, 2012, p. 503. URL: <http://doi.acm.org/10.1145/2207676.2207745> (cit. on p. 7).
- [29] Jonathan Hook et al. “A reconfigurable ferromagnetic input device”. In: *Proceedings of the 22nd annual ACM symposium on User interface software and technology - UIST '09*. ACM Press, 2009, p. 51. URL: <http://doi.acm.org/10.1145/1622176.1622186> (cit. on pp. 6, 32).
- [30] Thorsten Karrer et al. “Pinstripe: Eyes-free Continuous Input on Interactive Clothing”. In: *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. ACM Press, 2011, p. 1313. URL: <http://doi.acm.org/10.1145/1978942.1979137> (cit. on pp. 5, 10, 46).

- [31] Byron Lahey et al. “PaperPhone: Understanding the Use of Bend Gestures in Mobile Devices with Flexible Electronic Paper Displays”. In: *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. ACM Press, May 2011, p. 1303. URL: <http://doi.acm.org/10.1145/1978942.1979136> (cit. on p. 6).
- [32] Sang-Su Lee et al. “How Users Manipulate Deformable Displays As Input Devices”. In: *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, 2010, p. 1647. URL: <http://doi.acm.org/10.1145/1753326.1753572> (cit. on pp. 5, 10, 11, 34, 37, 46).
- [33] Daniel Leithinger and Hiroshi Ishii. “Relief”. In: *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction - TEI '10*. ACM Press, 2010, p. 221. URL: <http://doi.acm.org/10.1145/1709886.1709928> (cit. on p. 6).
- [34] Joanne Leong et al. “proCover: Sensory Augmentation of Prosthetic Limbs Using Smart Textile Covers”. In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. ACM Press, 2016, p. 335. URL: <http://doi.acm.org/10.1145/2984511.2984572> (cit. on pp. 14, 15).
- [35] Lin Shu, Xiaoming Tao, and David Dagan Feng. “A New Approach for Readout of Resistive Sensor Arrays for Wearable Electronic Applications”. *IEEE Sensors Journal* 15.1 (Jan. 2015), p. 442. URL: <https://doi.org/10.1109/JSEN.2014.2333518> (cit. on p. 19).
- [36] Fabien Lotte et al. “A review of classification algorithms for EEG-based brain-computer interfaces”. *Journal of neural engineering* 4.2 (2007). URL: <https://doi.org/10.1088/1741-2560/4/2/R01> (cit. on p. 24).
- [37] Dinesh Mandalapu and Sriram Subramanian. “Exploring Pressure As an Alternative to Multi-touch Based Interaction”. In: *Proceedings of the 3rd International Conference on Human Computer Interaction - IndiaHCI '11*. c. ACM Press, 2011, p. 88. URL: <http://doi.acm.org/10.1145/2407796.2407810> (cit. on p. 6).
- [38] David C. McCallum et al. “PressureText: Pressure Input for Mobile Phone Text Entry”. In: *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems - CHI EA '09*. ACM Press, 2009, p. 4519. URL: <http://doi.acm.org/10.1145/1520340.1520693> (cit. on p. 6).
- [39] Sachi Mizobuchi et al. “Making an Impression: Force-controlled Pen Input for Handheld Devices”. In: *CHI '05 extended abstracts on Human factors in computing systems - CHI '05*. ACM Press, 2005, p. 1661. URL: <http://doi.acm.org/10.1145/1056808.1056991> (cit. on p. 6).
- [40] Uran Oh and Leah Findlater. “Design of and Subjective Response to On-body Input for People with Visual Impairments”. In: *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility - ASSETS '14*. ACM Press, 2014, p. 115. URL: <http://doi.acm.org/10.1145/2661334.2661376> (cit. on p. 4).

- [41] Simon Olberding et al. “AugmentedForearm: Exploring the Design Space of a Display-enhanced Forearm”. In: *Proceedings of the 4th Augmented Human International Conference on - AH '13*. ACM Press, 2013, p. 9. URL: <http://doi.acm.org/10.1145/2459236.2459239> (cit. on pp. 36, 46).
- [42] Patrick Parzer et al. “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*. ACM Press, 2016, p. 3754. URL: <http://doi.acm.org/10.1145/2851581.2890253> (cit. on pp. 14, 15, 20).
- [43] Jerome Pasquero, Scott J. Stobbe, and Noel Stonehouse. “A Haptic Wristwatch for Eyes-free Interactions”. In: *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. ACM Press, 2011, p. 3257. URL: <http://doi.acm.org/10.1145/1978942.1979425> (cit. on p. 7).
- [44] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. *The Journal of Machine Learning Research* 12 (2011), p. 2825. URL: <http://dl.acm.org/citation.cfm?id=1953048.2078195> (cit. on p. 31).
- [45] Ivan Poupyrev et al. “Project Jacquard: Interactive Digital Textiles at Scale”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, 2016, p. 4216. URL: <http://doi.acm.org/10.1145/2858036.2858176> (cit. on p. 2).
- [46] Halley P. Profita et al. “Don’t Mind Me Touching My Wrist: A Case Study of Interacting with On-body Technology in Public”. In: *Proceedings of the 17th annual international symposium on International symposium on wearable computers - ISWC '13*. ACM Press, 2013, p. 89. URL: <http://doi.acm.org/10.1145/2493988.2494331> (cit. on p. 4).
- [47] Philip Quinn and Andy Cockburn. “Zoofing!: Faster List Selections with Pressure-zoom-flick-scrolling”. In: *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group on Design: Open 24/7 - OZCHI '09*. ACM Press, 2009, p. 185. URL: <http://doi.acm.org/10.1145/1738826.1738856> (cit. on p. 6).
- [48] Raf Ramakers et al. “RetroFab: A Design Tool for Retrofitting Physical Interfaces Using Actuators, Sensors and 3D Printing”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, 2016, p. 409. URL: <http://doi.acm.org/10.1145/2858036.2858485> (cit. on p. 12).
- [49] Gonzalo Ramos and Ravin Balakrishnan. “Zliding”. In: *Proceedings of the 18th annual ACM symposium on User interface software and technology - UIST '05*. ACM Press, 2005, p. 143. URL: <http://doi.acm.org/10.1145/1095034.1095059> (cit. on p. 6).
- [50] Gonzalo Ramos, Matthew Boulos, and Ravin Balakrishnan. “Pressure Widgets”. In: *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*. Vol. 6. 1. ACM Press, 2004, p. 487. URL: <http://doi.acm.org/10.1145/985692.985754> (cit. on p. 6).

- [51] Sebastian Raschka. *Mlxend*. Apr. 2016. URL: <http://dx.doi.org/10.5281/zenodo.49235> (cit. on p. 27).
- [52] Christian Rendl et al. “Presstures: Exploring Pressure-sensitive Multi-touch Gestures on Trackpads”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '14*. ACM, 2014, p. 431. URL: <http://doi.acm.org/10.1145/2556288.2557146> (cit. on pp. 6, 44).
- [53] Ilya Rosenberg and Ken Perlin. “The UnMousePad: An Interpolating Multi-touch Force-sensing Input Pad”. *ACM Trans. Graph.* 28.3 (July 2009), p. 651. URL: <http://doi.acm.org/10.1145/1531326.1531371> (cit. on pp. 6, 32).
- [54] T. Scott Saponas, Chris Harrison, and Hrvoje Benko. “PocketTouch: Through-fabric Capacitive Touch Input”. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. Figure 1. ACM Press, 2011, p. 303. URL: <http://doi.acm.org/10.1145/2047196.2047235> (cit. on p. 5).
- [55] T. Scott Saponas et al. “Enabling Always-available Input with Muscle-computer Interfaces”. In: *Proceedings of the 22nd annual ACM symposium on User interface software and technology - UIST '09*. ACM Press, 2009, p. 167. URL: <http://doi.acm.org/10.1145/1622176.1622208> (cit. on p. 7).
- [56] Toshiki Sato et al. “PhotoelasticTouch: Transparent Rubbery Tangible Interface Using an LCD and Photoelasticity”. In: *Proceedings of the 22nd annual ACM symposium on User interface software and technology - UIST '09*. ACM Press, 2009, p. 43. URL: <http://doi.acm.org/10.1145/1622176.1622185> (cit. on pp. 6, 32).
- [57] Mika Satomi and Hannah Perner-Wilson. *How To Get What You Want*. 2016. URL: <http://www.kobakant.at/DIY/?cat=24> (visited on 09/23/2015) (cit. on p. 2).
- [58] R. S. Saxena, R. K. Bhan, and Anita Aggrawal. “A new discrete circuit for readout of resistive sensor arrays”. *Sensors and Actuators, A: Physical* 149.1 (2009), p. 93. URL: <https://doi.org/10.1016/j.sna.2008.10.013> (cit. on p. 19).
- [59] Raghvendra Sahai Saxena, Navneet Kaur Saini, and R. K. Bhan. “Analysis of crosstalk in networked arrays of resistive sensors”. *IEEE Sensors Journal* 11.4 (2011), p. 920. URL: <https://doi.org/10.1109/JSEN.2010.2063699> (cit. on p. 18).
- [60] Raghvendra Sahai Saxena et al. “Virtual Ground Technique for Crosstalk Suppression in Networked Resistive Sensors”. English. *IEEE Sensors Journal* 11.2 (Feb. 2011), p. 432. URL: <https://doi.org/10.1109/JSEN.2010.2060186> (cit. on p. 19).
- [61] 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*. ACM Press, 2016, p. 108. URL: <http://doi.acm.org/10.1145/2971763.2971797> (cit. on pp. 5, 46).
- [62] Carsten Schwesig, Ivan Poupyrev, and Eijiro Mori. “Gummi: User Interface for Deformable Computers”. In: *Extended Abstracts on Human Factors in Computing Systems - CHI EA '03*. ACM Press, 2003, p. 954. URL: <http://doi.acm.org/10.1145/765891.766091> (cit. on p. 6).

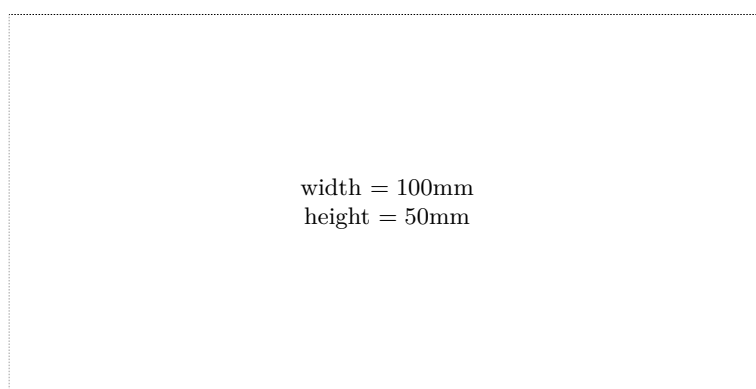
- [63] M. Shimojo, M. Ishikawa, and K. Kanaya. “A flexible high resolution tactile imager with video signal output”. In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1991, p. 384. URL: <http://ieeexplore.ieee.org/document/131607/> (cit. on p. 19).
- [64] Craig Stewart et al. “Characteristics of pressure-based input for mobile devices”. In: *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, 2010, p. 801. URL: <http://doi.acm.org/10.1145/1753326.1753444> (cit. on p. 6).
- [65] Mathias Sundholm et al. “Smart-mat”. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct*. ACM Press, 2014, p. 373. URL: <http://dl.acm.org/citation.cfm?doid=2632048.2636088> (cit. on p. 8).
- [66] Stuart Taylor et al. “Type-hover-swipe in 96 Bytes: A Motion Sensing Mechanical Keyboard”. In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, 2014, p. 1695. URL: <http://doi.acm.org/10.1145/2556288.2557030> (cit. on p. 29).
- [67] Giovanni Maria Troiano, Esben Warming Pedersen, and Kasper Hornbæk. “User-defined Gestures for Elastic, Deformable Displays”. In: *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces - AVI '14*. ACM Press, 2014, p. 1. URL: <http://doi.acm.org/10.1145/2598153.2598184> (cit. on pp. 5, 10, 11, 34, 46).
- [68] Nirzaree Vadgama and Jürgen Steimle. “Flexy: Shape-Customizable, Single-Layer, Inkjet Printable Patterns for 1D and 2D Flex Sensing”. In: *Proceedings of the Tenth International Conference on Tangible, Embedded, and Embodied Interaction - TEI '17*. ACM Press, 2017, p. 153. URL: <http://doi.acm.org/10.1145/3024969.3024989> (cit. on p. 46).
- [69] Anita Vogl et al. “StretchEBand: Enabling Fabric-based Interactions Through Rapid Fabrication of Textile Stretch Sensors”. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, 2017, p. 2617. URL: <http://doi.acm.org/10.1145/3025453.3025938> (cit. on p. 5).
- [70] Anita Vogl et al. “Understanding the everyday use of head-worn computers”. In: *2015 8th International Conference on Human System Interaction (HSI)*. IEEE, June 2015, p. 213. URL: <https://doi.org/10.1109/HSI.2015.7170668> (cit. on p. 12).
- [71] Julie Wagner et al. “Body-centric Design Space for Multi-surface Interaction”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, 2013, p. 1299. URL: <http://doi.acm.org/10.1145/2470654.2466170> (cit. on p. 4).
- [72] Martin Weigel, Vikram Mehta, and Jürgen Steimle. In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, 2014, p. 179. URL: <http://doi.acm.org/10.1145/2556288.2557239> (cit. on pp. 4, 5, 9, 12, 34, 46).

- [73] Martin Weigel et al. “iSkin: Flexible, Stretchable and Visually Customizable On-Body Touch Sensors for Mobile Computing”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. ACM Press, 2015, p. 2991. URL: <http://doi.acm.org/10.1145/2702123.2702391> (cit. on pp. 7, 46).
- [74] Mark Weiser. “Human-computer Interaction”. In: ed. by Ronald M. Baecker et al. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. Chap. The Computer for the 21st Century, p. 933. URL: <http://dl.acm.org/citation.cfm?id=212925.213017> (cit. on p. 1).
- [75] G Wilson. “Using Pressure Input and Thermal Feedback to Broaden Haptic Interaction with Mobile Devices”. In: *PhD Thesis*. June. 2013, p. 1. URL: <http://theses.gla.ac.uk/id/eprint/4363> (cit. on p. 6).
- [76] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. “User-defined gestures for surface computing”. In: *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*. ACM Press, 2009, p. 1083. URL: <http://doi.acm.org/10.1145/1518701.1518866> (cit. on pp. 9, 11, 35, 46).
- [77] Bo Zhou et al. “Never skip leg day: A novel wearable approach to monitoring gym leg exercises”. In: *IEEE International Conference on Pervasive Computing and Communications - PerCom '16*. IEEE, Mar. 2016, p. 1. URL: <https://doi.org/10.1109/PERCOM.2016.7456520> (cit. on p. 8).
- [78] Bo Zhou et al. “Smart Soccer Shoe: Monitoring Foot-ball Interaction with Shoe Integrated Textile Pressure Sensor Matrix”. In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers - ISWC '16*. Figure 1. ACM Press, 2016, p. 64. URL: <http://doi.acm.org/10.1145/2971763.2971784> (cit. on p. 8).



# Check Final Print Size

— Check final print size! —



— Remove this page after printing! —