Use the Force:
How Force Touch Improves Input on Handheld Touchscreens

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Christian Leonhard Maria Corsten, M.Sc.
aus Sittard, Niederlande

Berichter: Prof. Dr. Jan Borchers
Prof. Dr. Stephen Brewster

Tag der mündlichen Prüfung: 10. März 2020

Eidesstattliche Erklärung

Ich, *Christian Leonhard Maria Corsten*

erklärt hiermit, dass diese Dissertation und die darin dargelegten Inhalte die eigenen sind und selbstständig, als Ergebnis der eigenen originären Forschung, generiert wurden.

Hiermit erkläre ich an Eides statt

1. Diese Arbeit wurde vollständig oder größtenteils in der Phase als Doktorand dieser Fakultät und Universität angefertigt;

2. Sofern irgendein Bestandteil dieser Dissertation zuvor für einen akademischen Abschluss oder eine andere Qualifikation an dieser oder einer anderen Institution verwendet wurde, wurde dies klar angezeigt;

3. Wenn immer andere eigene- oder Veröffentlichungen Dritter herangezogen wurden, wurden diese klar benannt;

4. Wenn aus anderen eigenen- oder Veröffentlichungen Dritter zitiert wurde, wurde stets die Quelle hierfür angegeben. Diese Dissertation ist vollständig meine eigene Arbeit, mit der Ausnahme solcher Zitate;

5. Alle wesentlichen Quellen von Unterstützung wurden benannt;

6. Wenn immer ein Teil dieser Dissertation auf der Zusammenarbeit mit anderen basiert, wurde von mir klar gekennzeichnet, was von anderen und was von mir selbst erarbeitet wurde;

7. Ein Teil oder Teile dieser Arbeit wurden zuvor veröffentlicht und zwar in:

   ***Corsten, C.**, Lahaye, M., Borchers, J., Voelker, S.: ForceRay: Extending Thumb Reach via Force Input Stabilizes Device Grip for Mobile Touch Input. In Proc. ACM CHI '19.*

   ***Corsten, C.**, Voelker, S., Link, A., Borchers, J.: Use the Force Picker, Luke: Space-Efficient Value Input on Force-Sensitive Mobile Touchscreens. In Proc. ACM CHI '18.*

   ***Corsten, C.**, Voelker, S., Borchers, J.: Release, Don't Wait!: Reliable Force Input Confirmation with Quick Release. In Proc. ACM ISS '17.*

   ***Corsten, C.**, Daehlmann, B., Voelker, S., Borchers, J.: BackXPress: Using Back-of-Device Finger Pressure to Augment Touchscreen Input on Smartphones. In Proc. ACM CHI '17.*

5. Mai 2020 _____

Contents

Abstract **xxi**

Überblick **xxiii**

Acknowledgements **xxv**

Conventions **xxvii**

1 Introduction **1**

1.1 Hypotheses 8

1.2 Thesis Structure 9

2 Application and Sensing of Force **11**

2.1 What is Force? 11

2.1.1 Force vs. Pressure 12

2.1.2 Measuring Force with a Force Meter . . 13

2.2 Applying Force with Hands and Fingers 14

2.2.1 The Skeleton of the Hand 15

2.2.2 Joints, Tendons, and Muscles 15
2.2.3 Isotonic and Isometric Muscle Contraction 16
2.3 The Psychophysics of Force Input 18
2.3.1 Maximum Voluntary Contraction (MVC) 18
2.3.2 Discrimination of Force Magnitudes . . 19
2.3.3 Controlling Force with Precision 20
2.3.4 The Influence of Feedback 21
Internal Feedback 22
External Feedback 23
2.3.5 The Influence of Motion 25
2.4 Sensing and Digitizing Force 26
2.4.1 Force Sensing Resistors (FSRs) 27
2.4.2 Capacitive Force Sensing 30
2.4.3 Piezoelectric Force Sensing 32
2.4.4 Finger Contact Point 33
2.4.5 Finger Contact Size 34
2.4.6 Time-Based Force Estimation 35
2.4.7 Vibration-Based Force Estimation 36
2.4.8 Acoustics-Based Force Estimation 37
2.4.9 Pressure-Based Force Estimation 38
2.4.10 Light-Based Force Estimation 38

3 Related Work: Force-Based Interaction 41

3.1 Pressure-Based Linear Targeting—A Basic Force Input Technique 41

3.1.1 Hysteresis 42

3.1.2 Selection and Confirmation 43

Thresholding 44

Dwell Time 45

Quick Release 45

3.1.3 Transfer Function 46

3.1.4 Control Mechanisms 47

Position-Based Control 48

Rate-Based Control 48

3.2 Force-Based Interaction Techniques 50

3.2.1 Stationary Force Input Techniques . . . 50

Computer Mice 50

Touchscreens, Tablets, and Pads 52

Styli on Tablets 56

Isometric Joysticks 60

3.2.2 Handheld Force Input Techniques 64

Secondary “Click” 64

3.2.3 Text Input 67

Authentication 68

3D Object Manipulation 68
Grip Maintenance 69
Scrolling and Zooming 70
Gain Factor Control 72
3.3 Overview: Force Input in HCI 73

4 Adding Expressiveness to Handheld Touch Input with Back-of-Device Finger Force 85

4.1 Motivation 86
4.2 Related Work 88
4.2.1 Back-of-Device Interaction (BoDI) . . . 88
4.2.2 Force Input 89
Transfer Function and Selection 90
Transient Force 91
4.3 The BackXPress Interaction Technique 91
4.3.1 Application Examples 92
Text Input 92
Gaming 93
Hotel Finding 93
4.4 Study 1: BoD Force Finger Candidates 93
Results 94
4.5 Study 2: BoD Force Finger Pose 95
Results 97

4.6 Study 3: Force Dynamics and Tapping Accuracy 98

4.6.1 Experimental Design 99

Apparatus 99

Participants 99

4.6.2 Task 1: 1-Tap 99

Variables 101

4.6.3 Results – 1-Tap 102

Baseline Data 102

BoD Force Data 103

4.6.4 Discussion – 1-Tap 105

4.6.5 Task 2: N-Tap 106

Variables 107

4.6.6 Results – N-Tap 107

Baseline 108

BoD Force Data 108

4.6.7 Discussion – N-Tap 110

4.7 Improving Dynamic Force Control 112

4.8 Design Guidelines 114

4.9 Limitations and Future Work 115

4.10 Conclusion 116

5 Designing an Efficient Confirmation Technique for Handheld Force Touch Input 119

5.1 Motivation 120
5.2 Related Work 121
5.3 Detecting Quick Release 123
5.3.1 Experimental Design 123
5.3.2 Results 125
5.4 Verification Experiment 128
5.4.1 Results 129
5.4.2 Discussion 132
5.5 Limitations and Future Work 132
5.6 Conclusion 133

6 Extending Thumb Reach on Handheld Devices via Force Touch Input 135

6.1 Motivation 136
6.2 Related Work 138
6.2.1 Reachability Techniques 138
6.2.2 Force Input Techniques 140
6.3 The ForceRay Interaction Technique 141
6.4 Study 1: Reachability Techniques 143
6.4.1 Apparatus, Techniques, and Task 143
Techniques 144
Targets 145
Variables 147

6.4.2 Results 148

6.4.3 Discussion 151

6.5 Study 2: Trained User Performance 154

6.5.1 Apparatus and Task 154

6.5.2 Results 155

6.5.3 Discussion 155

6.6 Guidelines 159

6.7 Limitations and Future Work 159

6.8 Summary and Conclusion 161

7 Making Value Selection on Handheld Devices Efficient via Force Touch Input 163

7.1 Motivation 164

7.2 Related Work 166

7.2.1 Positional Control (PC) 167

7.2.2 Rate-Based Control (RC) 168

7.3 Bidirectional Force Input Techniques 169

7.4 Study 1: Force Picker vs. System Picker 172

Picker Design 172

Transfer Function and Force Pickers . . 174

7.4.1 Results 176

7.4.2 Discussion 181

7.5 Study 2: Minimizing the Force Picker 182

7.5.1 Results 183
7.5.2 Discussion 184
7.6 Study 3: Trained User Performance 185
7.6.1 Results 186
7.6.2 Discussion 187
7.7 Application Examples 189
Space-Efficient In-Row Value Selection . 189
Parameterized Shortcuts 189
Self-Paced Browsing 190
7.8 Limitations and Future Work 191
7.9 Conclusion 192

8 Summary, Future Perspectives, and Closing Remarks 195

8.1 Summary and Conclusion 195
8.1.1 Contributions and Benefits 199
8.2 Future Perspectives 200
8.3 Closing Remarks 201

Bibliography 203

Index 229

Own Publications 235

Papers 235

Book Chapters 238

Posters (peer-reviewed) 239

Posters (juried) 239

Theses 240

List of Figures and Tables

1.1 A user is taking a picture with her smartphone. 2

1.2 Squeezing a tube of toothpaste with different magnitudes of force. 3

1.3 Setup of a harpsichord musical instrument. . . . 4

1.4 Close-up of the hammer mechanism found in pianos. 5

1.5 Close-up of the IBM TrackPoint isometric joystick. 6

1.6 Spaceball input device by SiliconGraphics. . . . 7

2.1 Example of a force meter measurement. 14

2.2 Skeleton of the human hand. 16

2.3 Intrinsic hand muscles. 17

2.4 Extrinsic hand muscles. 17

2.5 Exploded assembly drawing of an FSR. 28

2.6 Exploded assembly drawing of a capacitive touch digitizer. 31

2.7 Exploded assembly drawing of an iPhone 6s touchscreen 33

3.1 PBLT binning example. 43

3.2 Force curves for confirmation via Dwell Time and Quick Release 47

3.3 Force-sensitive CRT touchscreen by Herot et al., 1978. 53

3.4 Pressure Widgets by G. Ramos et al. [2004]. . . . 57

3.5 Example of a Pressure Mark. 59

3.6 Example for a secondary click action. 65

3.7 Gain factor control via force input. 73

3.8 Force-based interaction techniques in HCI (1/9). 76

3.9 Force-based interaction techniques in HCI (2/9). 77

3.10 Force-based interaction techniques in HCI (3/9). 78

3.11 Force-based interaction techniques in HCI (4/9). 79

3.12 Force-based interaction techniques in HCI (5/9). 80

3.13 Force-based interaction techniques in HCI (6/9). 81

3.14 Force-based interaction techniques in HCI (7/9). 82

3.15 Force-based interaction techniques in HCI (8/9). 83

3.16 Force-based interaction techniques in HCI (9/9). 84

4.1 Emoji insertion with BackXPress. 87

4.2 Back-of-Device finger locations. 95

4.3 BackXPress: Findings from Study 1. 96

4.4 BackXPress: Findings from Study 2. 98

4.5 BackXPress: Handheld device used in Study 3. . 100

4.6 BackXPress: Time data for Study 3 (1-Tap). . . . 103

4.7 BackXPress: Force Range data from Study 3 (1-Tap). 104

4.8 BackXPress: Success data for Study 3 (1-Tap). . . 105

4.9 BackXPress: Force Range data from Study 3 (N-Tap). 109

4.10 BackXPress: Tap Count data for Study 3 (N-Tap). 109

4.11 BackXPress: Success data for Study 3 (N-Tap). . 111

4.12 BackXPress: Typical force curve for a 1-Tap trial from Study 3. 113

4.13 BackXPress: Typical 20 s force curve for an N-Tap trial from Study 3. 113

4.14 BackXPress: Improved force control success through force history data. 114

4.15 Bluetooth-enabled BackXPress hardware prototype with six FSRs. 116

5.1 Interaction stages for force input confirmed via Quick Release and Dwell Time. 121

5.2 Apparatus for Quick Release user study. 125

5.3 Quick Release: Typical plot of reversed frames for a force-matching task from one user. 126

5.4 Quick Release: Cumulated count of reversed frames from all trials matching the requested force level. 127

5.5 Quick Release: Success rates for different combinations of reversed frame and width. 128

5.6 Quick Release: Time data from the verification experiment. 130

5.7 Quick Release: Success data from the verification experiment. 131

6.1 Interaction steps of the ForceRay interaction technique. 137

6.2 ForceRay: Targets and distractors layout for Study 1. 146

6.3 ForceRay: Time and Success data from Study 1. 149

6.4 ForceRay: Time and Success data from Study 1 split by Size and Target. 149

6.5 ForceRay: Device Rotation data from Study 1. . 150

6.6 ForceRay: Device Rotation data from Study 1 split by Size and Target. 150

6.7 ForceRay: Gesture footprint from Study 1. . . . 152

6.8 ForceRay: Participants' Likert scale responses from Study 1. 153

6.9 ForceRay: Time and Success data from participants' training sessions from Study 2. 156

6.10 ForceRay: Time and Success data from Study 2. 156

6.11 ForceRay: Rotation data from Study 2. 157

6.12 ForceRay: Participants' Likert scale responses from Study 2. 157

6.13 Guidelines for picking the right reachability technique. 160

7.1 Entering a time value via a picker vs. the Force Picker on a smartphone. 165

7.2 Force curves for our Pulse, Press-Through, and Thumb-Roll techniques. 170

7.3 Force Picker, System Picker, and Minimized Thumb-Roll Picker. 173

7.4 iOS system picker and Android 7 system picker. 174

7.5 Force Picker: Picker travel distances for Study 1. 176

7.6 Force Picker: Time data for Study 1. 178

7.7 Force Picker: Crossings and Success data for Study 1. 178

7.8 Force Picker: Crossings data for Study 1 split by Range and Distance. 179

7.9 Force Picker: Gesture Footprint for Study 1. . . 180

7.10 Force Picker: Time and Success data for Study 2. 184

7.11 Force Picker: Gesture Footprint for Study 2. . . 185

7.12 Force Picker: Time per participant and training session for trained Force Picker use. 187

7.13 Force Picker: Time, Crossings, and Success data for Study 3. 187

7.14 Force Picker: Gesture footprint for Study 3. . . . 188

Abstract

Handheld devices, such as smartphones, have become essential tools in our everyday life. We use them, e.g., to contact people, browse the web, or take pictures. For whatever use, to interact with the handheld device, we hold it with one or two hands and touch with our fingers on the built-in touchscreen. However, this interaction is often constrained to simple contact between the finger and the flat display glass, although touch offers further, richer properties. One such rich property is the intensity of a touch, i.e., its *force*, that the user applies with every tap to the touchscreen. Incorporating this property into the user's interaction with the handheld device enables her to become more expressive with every single touch. In this thesis, we present a series of interaction techniques that take advantage of force touch input to make handheld interaction more efficient:

When holding the device with two hands in landscape orientation, most of the fingers are unavailable for interaction, since they rest at the back of the device (BoD), holding it in place. Using BoD force input, we can make efficient use of these fingers without sacrificing stability of the device grip. Our technique, *BackXPress*, enables quick access to shortcuts and menus to augment users' touch interaction with the frontal screen.

For single-handed device use, users can only use their thumb to interact with the frontal touchscreen but cannot reach everywhere without re-grasping the device. Our virtual thumb extension, *ForceRay*, lets the user cast a ray at unreachable targets and control a cursor on that ray that moves closer to these targets the more force is applied. The technique is ergonomic for the thumb and enables users to maintain a steady device grip. Targets located at the screen edges, like menus and navigation buttons, are acquired quickly.

Selection of values from long ordered lists, such as picking a date or time, can also be sped up when using force input. With our *Force Picker*, users scroll through the value range at various speeds, with the speed being coupled to the force exerted by the thumb. Prior rolling of the thumb to the left or right sets the scrolling direction. Compared to touch-based pickers, our Force Picker not only makes selection faster, but also only consumes little screen space since the gesture footprint for force input is much smaller.

While controlling force via fingers requires practice, we show that with training and algorithmic optimizations, users become quickly familiar with force input and gain the benefits of the added expressiveness for handheld interaction.

Überblick

Mobilgeräte mit Touchscreen, z.B. Smartphones, haben sich zu unseren Alltagswerkzeugen entwickelt. Wir kommunizieren, navigieren, oder machen Fotos mit ihnen. Unabhängig vom Nutzen interagieren wir mit dem Mobilgerät durch Berührung des Touchscreens. Diese Berührung mit dem Finger ist jedoch wenig reichhaltig; es wird lediglich die Position der Berührung zur Eingabe genutzt. Eine reichhaltigere Eigenschaft ist die Intensität der Berührung, d.h. mit welchem *Druck* der Finger auf den Touchscreen trifft.

Diese Doktorarbeit erforscht neue Techniken, die aufzeigen, wie die Druckeingabe dem Nutzer eine neugewonnene Ausdrucksstärke zur Verfügung stellt, mit der er effizienter mit Mobilgeräten interagieren kann.

Hält man das Gerät beidhändig quer, so können die meisten Finger nicht zur Interaktion genutzt werden, da sie auf der Geräterückseite aufliegen, um das Gerät festzuhalten. Wir stellen die Technik *BackXPress* vor, die es mittels Druckeingabe erlaubt diese Finger zur Steuerung von Menüs oder Kurzbefehlen zu nutzen, ohne die Grifffestigkeit einzubüßen.

Wenn eine einhändige Interaktion gewünscht ist, kann lediglich der Daumen auf der Vorderseite genutzt werden und dabei nicht einmal den gesamten Bereich des Touchscreens ohne Umgreifen des Geräts erreichen. Unsere virtuelle Daumen-Erweiterung *ForceRay* lässt den Nutzer entfernte Ziele auf dem Touchscreen mittels eines Cursors erreichen, der abhängig vom Daumendruck entlang der Erweiterung wandert. Insbesondere Menüs oder Navigations-Buttons am Rand des Gerätebildschirms sind auf diese Weise schnell und ergonomisch zu erreichen, während das Gerät dabei stabil in der Hand liegt.

Druckeingabe hilft ebenfalls Werte aus langen, geordneten Listen effizient auszuwählen. Mit unserem *Force Picker* iteriert der Nutzer durch den Wertebereich mit unterschiedlicher Geschwindigkeit, die mittels Daumendruck gesteuert wird. Ein vorheriges Rollen des Daumens legt die Iterationsrichtung fest. Verglichen mit konventioneller Touch-Eingabe spart unser Force Picker Zeit und Bildschirmplatz bei der Selektion der Werte.

Auch wenn sich Nutzer anfangs an die Druckeingabe gewöhnen müssen, so konnten wir in unseren Studien zeigen, dass Übung und der Einsatz von algorithmischen Optimierungen die Druckeingabe besser beherrschbar machen und unsere Nutzer so von der gewonnenen Ausdrucksstärke auf Mobilgeräten profitieren.

Acknowledgements

Writing a PhD thesis is not possible without the immense support of advisors, colleagues, thesis students, study participants, family, and friends.

First, I want to thank my first advisor, Prof. Jan Borchers, for giving me the opportunity to do a PhD thesis in *Human-Computer Interaction (HCI)* at *RWTH Aachen University*. During my years at the lab, I not only learned how to conduct research but also benefited from Jan's expertise and advise in writing, teaching, and critical thinking. Thank you for supporting my research ideas and projects and making the lab an enjoyable place to work at.

Second, I want to thank my second advisor, Prof. Stephen Brewster. Thank you for spending the time and effort to co-supervise my thesis. Your research about force input in HCI has truly inspired me and paved the way for my thesis.

Many thanks go to Nur Hamdan, Dr. Thorsten Karrer, Dr. Simon Voelker, and Assistant Prof. Chatchavan Wacharamanotham for intense and immensely valuable research discussions that shaped my way through my PhD career.

Of course, I want to thank all colleagues, student assistants, and thesis students that I have worked with. I am sorry that I cannot mention you all by name since the list would be too long. I really enjoyed working with you and learning from you.

Special thanks go to my lovely friends Mazhar, Alex, Gamze, Claire, Manar, and Eyadeh. All of you were always supportive and helped me recovering from stressful weekends of writing papers and my thesis. Thanks for your Syrian-Greek-Turkish-Jordan cultural inspirations, food, and exceptional warmth.

Last but not least, I want to thank my family, in particular my parents who supported me throughout my PhD, especially during stressful times. Without your help and understanding this thesis would not have been possible. I cannot thank you enough!

Thank you all for making this part of my life unforgettably awesome!

—Christian

Conventions

Throughout this thesis we use the following conventions:

- The thesis is written in American English.
- The first person is written in plural form.
- Unidentified third persons are described in female form.

Summaries, examples, or short excurses are set off in colored boxes.

> → Summary
>
> Summaries of own publications are set off in blue boxes.

> → Knowledge Applied
>
> Applied examples are set off in red boxes.

> → Excursus
>
> Excurses are set off in green boxes.

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

This is a summary of a paragraph.

Source code or implementation symbols are written in `typewriter-style text`.

1 Introduction

Handheld devices, such as tablets and smartphones, have become essential tools in people's everyday life. Especially smartphones are used in manifold contexts, since they are compact and can be easily pocketed, making them readily accessible [Stadd, 2013]. Exemplary use cases for the smartphone include texting family and friends, checking mails on the go, scheduling appointments, playing mobile games, watching videos, reading news while riding the bus, paying in shops, tracking steps and sports activities, booking an Uber ride, using the smartphone as navigation aid, or taking pictures [Davies, 2017] (Fig. 1.1). All these applications are typically operated by the user via the device's built-in touchscreen that serves for both, input and output. This collocation of input and output creates the illusion of directly interacting with objects on the screen, also known as *direct manipulation* [Shneiderman, 1997].

Handheld devices, such as smartphones, have become prevalent tools in people's everyday life. Through finger touch input that is sensed by the built-in touchscreen, smartphone users can text with other people, play games, or take pictures.

From an evolutionary point of view, using hands and fingers to interact with objects has always been the most natural way of human-object interaction [Kivell, 2015]. Using their hands, humans touch, grasp, or squeeze physical objects to interact with them. Typically, these manipulations include the application of force, i.e., the intensity with which hands and fingers touch, grasp, or squeeze the object. An apt example is the usage of a tube of toothpaste (Fig. 1.2): Grasped by the thumb, index-, and middle finger, one exerts gentle force with the thumb on the tube head to extrude a little bit of toothpaste onto the toothbrush. To add more toothpaste, one can either repeat this process—which is tedious—or simply increase the finger force while pressing

While touchscreens sense a user's touch input, they neglect a powerful and natural property that comes with every touch: its intensity, i.e, its *force*.

Figure 1.1: Handheld devices, such as smartphones, have become essential tools in everyday life, e.g., for taking pictures. Picture taken from `https://pxhere.com`.

against the tube. By varying the magnitude of force applied to the tube, one can extrude different amounts of toothpaste while always using the same fingers and grip. Hence, force makes "input" from fingers highly *expressive*.

Force makes a user's touch input more *expressive*. An example is the invention of the piano that enables players to control the loudness of a tone via finger force.

While the example above is representative for many other object manipulations via force from multiple fingers, already the force from a single finger gives humans high expressiveness at hand. One historical example that illustrates how force increases human's expressiveness with every finger movement is the invention of the piano [Wigdor et al., 2011]. The precursor of this keyboard music instrument is the harpsichord. Like the piano, the harpsichord consists of a set of keys that the player touches to play tones. Mechanically, each of the harpsichord keys plucks a string to generate a tone (Fig. 1.3). While the player can play individual tones through touching different keys, she cannot control the loudness of the tones other than playing multiple notes at the same time. To tackle this issue, the piano uses a hammer mechanism instead that lets the player individually control the loudness of each tone depending on the intensity with which each key is being touched (Fig. 1.4). Hitting a key stronger strikes the connected string harder such that the sound is louder. Hence, the more force the player exerts on a key, the louder the tone. This new degree of freedom was not only beneficial to the players of keyboard music but also led to the creation of new musical

Figure 1.2: Squeezing of a tube of toothpaste grasped by the thumb, index-, and middle finger. (a) Gently applying force with the thumb the tube extrudes a little bit of toothpaste. (b) The more force is exerted, the more toothpaste is being squeezed out of the tube, still using the same grip and fingers.

pieces by composers. Also modern, digital pianos sense the intensity of touch on each key. Instead of using a mechanical hammer mechanism, force is detected electronically by sensing the velocity with which the player strikes the key. In addition, dedicated force sensors enable the player to create an 'after touch' effect that enables the player to add effects to a tone while it is sounding.

Based on the context, the expressiveness of a gesture can change.

The comparison of the harpsichord and the piano is an apt example to explain the term *expressiveness*: While basically the same gesture, i.e., putting the finger on a key, is executed, in the context of the piano, the player has additional control over the loudness of the tone, compared to when executing the same gesture on the harpsichord. In other words: Pressing a key on the harpsichord at different intensities always results in the same level of volume, while executing the same gesture on the piano keyboard lets the player express many different volumes.

Also in *Human-Computer Interaction (HCI)*, force increases the expressiveness of a

However, music is not the only domain that exploits the expressiveness of force. Also input devices for graphical user interfaces in desktop computing systems make use of force to increase the user's expressiveness. An apt example of a force-sensitive input device is the *isometric joystick* [Selker et al., 1991]. The isomet-

Figure 1.3: Setup of a harpsichord. The harpsichord consists of a keyboard that is connected to a set of strings on top of a resonating body. Hitting a key plucks a string such that a tone is generated and amplified via the resonating body. The force with which the player hits the key has no effect on the loudness of the generated tone. Image (slightly cropped and resized) taken by Sergej Medvedev.

user's input. One example is the isometric joystick that lets the user control a cursor via force without physically moving her finger.

ric joystick consists of a rubber knob that the user pushes with her index finger to steer a mouse cursor on the computer screen (Fig. 1.5). Strain gauges capture the magnitude and the direction of the force applied to the knob. While the magnitude determines the cursor velocity, the direction tells the cursor where to move on the screen. Hence, the isometric joystick makes input from a single finger very expressive although the finger does not physically move.

Figure 1.4: Close-up of the hammer mechanism found in pianos. When hitting a key, the hammer strikes a string such that a tone is generated. Depending on how strongly the player hits the key, the hammer will pluck the string differently: the more force is exerted on a key, the louder the tone will sound. In comparison to the harpsichord, the piano increases the player's expressiveness since she can now also control the loudness of each tone. Image (slightly cropped and resized) taken by Sergej Medvedev.

The Spaceball is an isometric input device that enables the user to control 3D virtual objects via force.

A similar input device that makes use of force from multiple fingers is the *Spaceball* (Fig. 1.6). The Spaceball is based on a fixed sphere that allows the user to control six degrees of freedom, typically for interaction with 3D virtual objects. For example, using the Spaceball one can pan, zoom, and rotate a virtual object along the x-, y-, and z-axis. Similar to the isometric joystick, the Spaceball is made of strain gauges that capture the force magnitude and direction applied by the user.

On handheld devices, force input is rarely exploited.

Looking at interaction with handheld devices, however, the expressiveness of force input is mostly not exploited. Touchscreens are typically rigid, flat glass surfaces that that constrain interaction with virtual objects to two-dimensional, "flat" gestures, such as tapping or swiping, performed with the finger on that surface [Victor, 2011]. Yet, in 2015, Apple introduced *3D Touch* to their newest iPhone smartphone lineup. 3D Touch enables

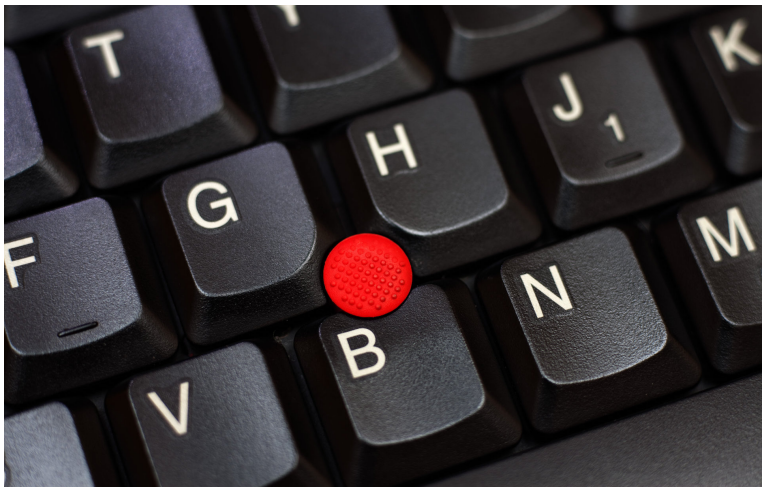


Figure 1.5: Close-up of an IBM TrackPoint isometric joystick added to a computer keyboard. The red rubber knob covers a set of strain gauges that capture the magnitude and the direction of force applied to it by a single finger. The joystick is used as pointing device to steer a cursor on a computer screen. The more force is applied, the faster the cursor moves. Image (slightly cropped and resized) taken by Ashley Pomeroy.

users to access shortcuts or previews by pressing with the finger against the touchscreen that continuously senses force input for every touch. Still, the potential of force is undermined since the expressiveness of 3D Touch is low, enabling the user to only express one or two additional states per force touch.

HCI has started looking into the potential of force input on handheld devices.

Research in *Human-Computer Interaction (HCI)* has also looked into force input for handheld device interaction. This includes single-finger scrolling [Antoine et al., 2017] and zooming [Miyaki et al., 2009] via force, using force as a secondary "click", e.g., to type capital letters [Brewster et al., 2009] on a touch keyboard, to enter pass codes on handheld devices that grant access when the code has been entered with a particular force [Arif et al., 2014], or to push 3D virtual objects along the z-dimension [Qiu et al., 2016]. However, these techniques typically require the user to apply force at a particular location on the handheld device—usually next to the touchscreen depending on where the force sensor is located—which violates the direct manipulation paradigm, or, like 3D Touch, they only add one or two additional states to the interaction.

Figure 1.6: Example of a Spaceball input device manufactured by SiliconGraphics. The ball at the center of the input device can be pushed, pulled, and twisted to control up to six degrees of freedom, e.g., to move and scale 3D virtual objects. Like the isometric joystick, the Spaceball consists of strain gauges that capture the direction and the magnitude of the force applied to the ball. The keys (1–8) are used for accessing UI shortcuts. Image (slightly modified and resized) taken by Rama.

Recent handheld devices bring continuous force-sensing to the entire display surface of the built-in touchscreen. This enables the design of new efficient interaction techniques for handheld devices.

With the advent of handheld touchscreens that bring continuous force-sensing to the entire display surface, we were interested in seeing whether the gained expressiveness from force input can make handheld interaction more efficient: Technically, at a single touch point, one can apply any of the possible forces that can be registered by the sensors. This has the same expressiveness as placing as many touch points along one dimension on the touchscreen that cannot sense force. Hence, force touch input consumes less space on the display compared to ordinary touch input, as the finger does not need to move across the touchscreen to yield the same expressiveness. We hypothesize that this unique

feature has the potential to benefit and increase the efficiency of handheld device interaction in several ways.

1.1 Hypotheses

H1 *Force input enables multi-finger input.* Usually, many fingers are located at the sides or at the back of the handheld device to hold it in place. Since force input brings high expressiveness to the static location of a finger, it can be used for input without moving the finger, hence, without sacrificing grip stability of the device.

H2 *Force input enables ergonomic one-handed use with input from a single finger.* When the device is held with one hand, the user can only touch the screen via the thumb. Typically, the thumb cannot reach the entire touchscreen to tap all targets without changing the device grip. Since force input requires less finger movement for the same expressiveness of touch input, it can help virtually reaching for otherwise unreachable areas on the screen.

H3 *Force input helps optimizing screen real estate.* For the same expressiveness, force input has a lower gesture footprint than touch input due to fewer finger displacement on the touchscreen. This typically preserved gesture space that collocates with the display space can thus be used otherwise, when relying on force input.

H4 *Force input enables fast selection of items or values.* Selection of a value on a touchscreen, such as setting a date or time, typically requires a lot of finger movement, e.g., to type on a numpad, drag a slider, or spin a picker. Force input could speed up value selection as no time-consuming finger displacement is required for selecting a value.

To verify these hypotheses, we designed and evaluated a set of force-based interaction techniques that we present in this thesis.

In particular, we present:

- *BackXPress*, an interaction technique that augments two-handed touch input for landscape-held smartphone use with force input from the fingers resting at the back of the device **(H1, H4)**,
- a reliable and efficient technique to confirm the selection of value input via force touch input for single-handed, single-finger device use **(H4)**,
- an interaction technique, called *ForceRay*, that brings ergonomic one-handed use to smartphones with large touchscreens while the user maintains a steady device grip **(H2)**, and
- a force-controlled widget, called *Force Picker*, that enables efficient value selection and use of screen space on handheld touchscreens that are operated by the thumb **(H3, H4)**.

1.2 Thesis Structure

The thesis is structured into eight chapters, each of which can also be consumed as a self-contained unit:

- **Chapter 2** gives an overview about force as a human-controllable input modality. This includes how force is defined and how it can be measured on handheld devices. Furthermore, the chapter discusses human capabilities and limitations of controlling force with hand and fingers.
- **Chapter 3** presents existing force-based interaction techniques designed for stationary and handheld input devices. An overview of 77 research papers that use force input for stationary and mobile interactions is presented.

- **Chapter 4, Chapter 5, Chapter 6,** and **Chapter 7** present the design and evaluation of our four force-based interaction techniques that address efficiency and expressiveness (H1–H4) for touch interaction with handheld devices.
- **Chapter 8** summarizes the thesis, draws a conclusion, and discusses future perspectives for force-based handheld interaction.

2

Application and Sensing of Force

To understand how force can be utilized to design interaction techniques, we need to consider both, the *human* who *applies* the force and the *technology* that *senses* the force. To start with, we give a brief definition of force as defined in physics. Then, we take a closer look at the human hand to understand what muscles and joints are involved in the application of force via hand and fingers. In this context we also discuss human factors, i.e., capabilities and limitations with respect to the execution of force, as well as influences by feedback technologies and motion on human force control. Next, we consider technological aspects, i.e., what techniques exist to sense and digitize force, especially for the use in handheld devices. This basis will help understanding how to design force-based interaction techniques that are both, controllable by the human, and measurable by the device.

In this chapter, we discuss how humans can control force with hand and fingers and what factors, such as feedback technologies, influence human force control.

2.1 What is Force?

In physics, the term *force* is defined by *Isaac Newton's Three Laws of Motion*. The first law of motion addresses the inertia of a body or object: It postulates that a body is either at rest or it is moving across a straight line at constant motion as long as either no force is acting on the body or the resulting force of all acting forces is equal to zero. Consequently, if a body changes its state of motion,

In physics, *force* is basically defined as the product of the mass of a body, e.g., an object, and its acceleration.

i.e., it either accelerates or decelerates, then this is caused by a force. The second law of motion puts force and acceleration into context: The produced acceleration on a body is proportional to the magnitude and the direction of the force, which is formulated by

$$F = m \cdot a \tag{2.1}$$

where m is the mass of the body or object, usually measured in kilograms (kg), and a stands for its acceleration, hence a constant change in motion that is measured in meters per second per second ($\frac{m}{s^2}$). Hence, the force F is measured in $kg \cdot \frac{m}{s^2}$, which is also defined as a Newton (N). The third law of motion tells that when a body exerts force on a second body, e.g., a finger pressing against a touchscreen, then the second body (the touchscreen) exerts an equal force on the first body (the finger) in the opposite direction. This action-reaction principle explains why a body deforms or bends, like the pad of the finger on the glass of the touchscreen.

Forces are classified as either *contact forces* or *contactless forces*.

Physics distinguishes *contactless forces*, such as the electromagnetic force, from *contact forces*. In the scope of this thesis, we deal with the latter, i.e., when two bodies exert force on each other while being in direct contact to each other. In particular, we deal with so-called *applied force*, i.e., muscular force that is applied onto a body. An apt example for applied force is the force exerted when pressing with a finger against a touchscreen.

2.1.1 Force vs. Pressure

In contrast to force, *pressure* is defined as force per area.

Note that, although the terms *force* and *pressure* are often used interchangeably in the literature, they have different meanings. In contrast to force, pressure (P) is defined as force per area (A):

$$P = \frac{F}{A} \tag{2.2}$$

Pressure is measured in Pascal: $1\ pa = 1\ \frac{N}{m^2}$. The definition of pressure entails that when the same force is applied with different finger pad sizes, like index finger and thumb, then the pressure will have different magnitudes. Also note that pressure is usually associated with liquids and gases rather than solid bodies. Since we solely deal with forces between solid bodies and and we strive for a uniform, comparable measure, we use the

term ‘force’ throughout this thesis. Still, any mention of the word ‘pressure’, e.g., in HCI literature should be considered a synonym to the meaning of the word ‘force’.

2.1.2 Measuring Force with a Force Meter

Force can be measured mechanically with a spring-loaded *force meter*.

A classic, i.e., mechanic, way of measuring force, is by using a *force meter*. A force meter, also referred to as *dynamometer*, is a measurement tool that consists of a mechanical spring that is connected to a hook. When a body is attached to the hook or when the hook is being pulled, the spring extends from its resting position by a certain distance, x (Fig. 2.1). According to *Hooke's Law*, the force F required to stretch the spring is proportional to the distance x:

$$F = k \cdot x, \tag{2.3}$$

where k is a constant (measured in $\frac{N}{m}$) that is characteristic for the spring used, and x (measured in m) is the distance by which the spring elongated from its resting state. Hence, when the spring characteristic k is known and x is measured, then F can be determined.

Force and *restoring force* oppose each other and to maintain an equilibrium.

Strictly speaking, when the spring is being pulled, two forces are exerted on the spring that oppose each other and maintain an equilibrium, i.e., the spring keeps its elongated distance x as long as it is being pulled with constant force F. When the pulling force F is released, the spring will go back to its resting state. This is caused by the *restoring force* $F_R = -k \cdot x$ that has the same magnitude as F but with opposed direction. While a force meter is typically used to measure a *pulling force*, one could also push the hook to compress the spring, which results in a negative value for x, hence a force magnitude for F and a positive force for F_R.

To sense force input on handheld devices, using a force meter is impractical.

The force meter comes with an analog display that shows the magnitude of F in Newton. While the force meter is a convenient measurement tool for physical experiments, it is impractical for our use case, i.e., for measuring force applied to a handheld device. Before looking into different techniques that allow sensing and digitizing force for the use in handheld devices, we will take a closer look at how humans actually apply force using hands and fingers.

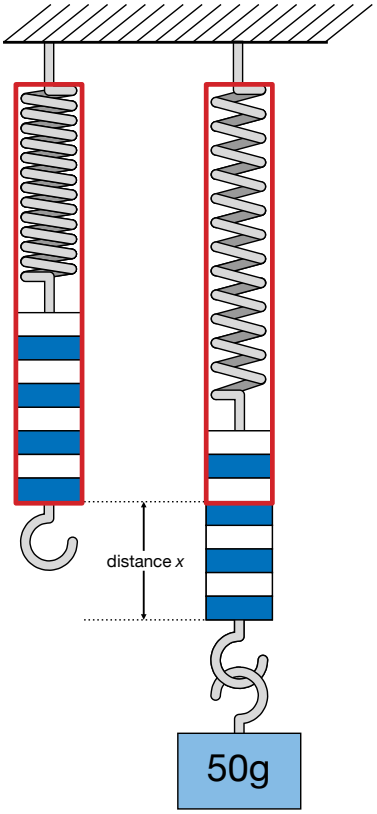


Figure 2.1: Example of a spring-loaded force meter to determine the pulling force induced by a mass of 50 g. Left: The force meter is in its resting position. Right: Adding a mass of 50 g causes the spring to be pulled down by distance x due to gravity. Knowing the spring characteristic k, the the pulling force F can be determined by multiplying k with x (*Hooke's Law*).

2.2 Applying Force with Hands and Fingers

Subsequently, we give a short overview about phalanges, joints, and muscles involved in the application of force via hand and fingers.

In this section, we cover the anatomy of the hand and fingers and, in particular, give a brief overview of bones, joints, and muscles involved in the application of force. This includes the constellation of phalanges and the flexing of their connecting joints to *extrinsic muscles* that are located in the lower arm and *intrinsic muscles* that are located in the hand. Please note that this section rather serves as an overview about how humans can apply force

with their fingers. More detailed information about this topic can be found in Further information can be found in, e.g., Hirt et al., 2017 and OpenStax College, 2013. In particular, we do not cover the sensation of force via muscle and skin receptors.

2.2.1 The Skeleton of the Hand

Each finger consists of two or three bones, called *phalanges*.

Figure 2.2 shows the skeleton of the hand with individual fingers. Each finger, except the thumb, consists of three *phalanges*: the *Distal Phalanx*, the *Middle Phalanx*, and the *Proximal Phalanx*. The Distal Phalanx builds the fingertip, whereas the Proximal Phalanx is adjacent to the bones of the palm, called *Metacarpal Bones*. The Middle Phalanx is located between the Distal Phalanx and the Proximal Phalanx. Only for the thumb, the Middle Phalanx is inexistent. The Metacarpal Bones reach until the wrist and are adjacent to the *Carpal Bones*. The Carpal Bones bound the *Carpal Tunnel*, which is a passage through which nerves and tendons pass from the hand to the lower arm.

2.2.2 Joints, Tendons, and Muscles

The phalanges are connected by the *DIP*, *PIP*, and *MCP* joints.

Joints connect adjacent phalanges. The joint between the Distal Phalanx and the Middle Phalanx is called the *Distal Interphalangeal Joint (DIP)*, the joint between the Middle Phalanx and the Proximal Phalanx is called the *Proximal Interphalangeal Joint (PIP)*, and the joint between the Proximal Phalanx and the Metacarpal Bones is called the *Metacarpophalangeal Joint (MCP)*.

The hand muscles are grouped into *extrinsic* and *intrinsic* muscles.

Muscles connect to these joints via tendons in oder to flex the phalanges. The hand muscles are grouped into *extrinsic* and *intrinsic* muscles. Extrinsic hand muscles are located in the lower arm and are responsible for flexing the DIP and PIP joints, whereas intrinsic muscles are located in the hand between the Metacarpal Bones and are generally responsible for lateral movement of the fingers. Figure 2.3 shows the different intrinsic hand muscles and their location: The three *Palmar Interossei Muscles* and the four *Dorsal Interossei Muscles* are located between and on top of the Metacarpal Bones and enable abduction and adduction of the fingers. Abduction and flexion of the little finger is controlled by the three *Hypothenar Muscles* that are located around

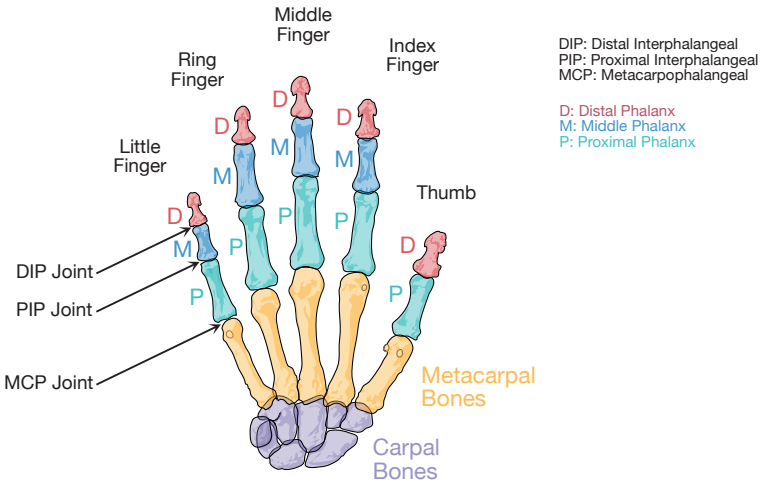


Figure 2.2: Skeleton of the human hand. The finger bones are called *phalanges* and are classified into *Distal Phalanx*, *Middle Phalanx*, and *Proximal Phalanx*. Graphic (slightly extended) by Mariana Ruiz Villarreal.

the Metacarpal Bone from the little finger. The three *Thenar Muscles* are responsible for flexion of the thumb towards opposing fingers and the four *Lumbricalis Muscles* enable flexing of each finger's MCP and thus are located between the fingers.

Extrinsic hand muscles located in the lower arm are responsible for flexion the DIP and PIP joints of each finger.

Figure 2.4 illustrates the extrinsic muscles for flexion of the thumb and DIP and PIP joints of the fingers. The *Flexor Pollicis Longus* connects to one tendon that connects to the Distal Phalanx of the thumb to allow flexion of the thumb's DIP. Flexion of the fingers' DIP happens via the *Flexor Digitorium Profundus* that connects via four tendons to each Distal Phalanx. The *Flexor Digitorum Superficialis* is responsible for flexing the PIP of each finger and connects via four tendons to each Proximal Phalanx.

2.2.3 Isotonic and Isometric Muscle Contraction

Muscle contraction is either *isotonic* or *isometric*.

Both, intrinsic and extrinsic muscles are involved in the application of force via fingers. In general, the contraction of muscles can be classified as *isotonic* or *isometric*. Isotonic contractions are characterized by a change in the length of the muscle while its tension is constant. Such contraction is typical for spatial move-

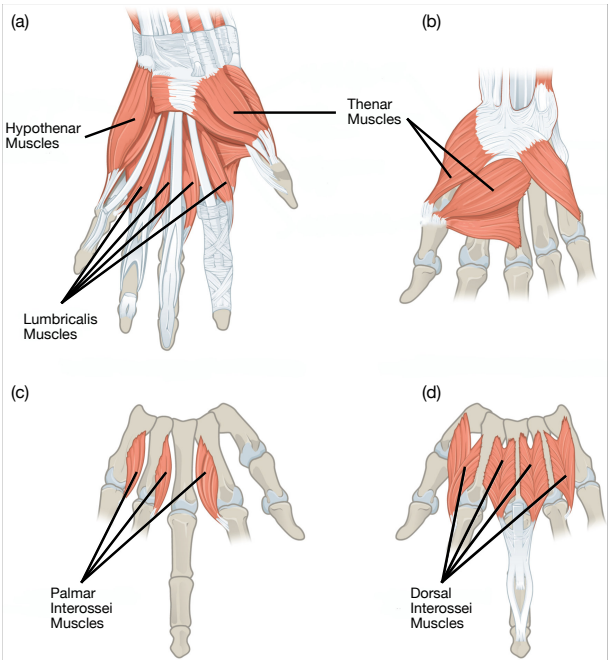


Figure 2.3: Intrinsic hand muscles of the human hand involved in the exertion of force with hand and fingers. (a) Palmar view of *Hypothenar-*, *Thenar-*, and *Lumbricalis Muscles*. (b) Dorsal view of *Thenar Muscles*. (c) Palmar view of *Interossei Muscles*. (d) Dorsal view of *Interossei Muscles*. Graphic (slightly modified) taken from OpenStax College, 2013.

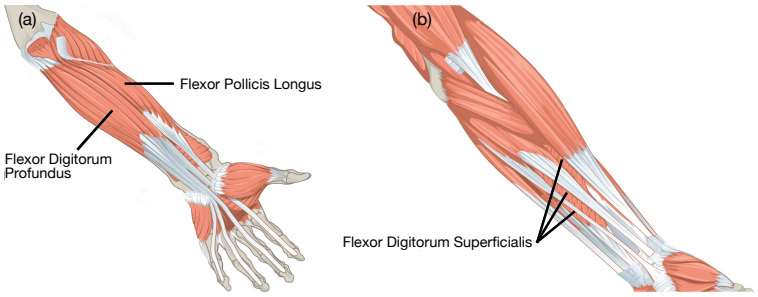


Figure 2.4: Extrinsic hand muscles of the human hand involved in the flexion of the thumb and DIP and PIP joints of the fingers. (a) Palmar view of the *Flexor Pollicis Longus* and *Flexor Digitorum Profundus*. (b) Palmar view of the *Flexor Digitorum Superficialis*. Graphic (slightly modified) taken from OpenStax College, 2013.

ments, e.g., when moving a computer mouse to steer a cursor, or performing drag-and-swipe gestures on a touchscreen. From a physiological point of view, an isotonic contraction means that the intrinsic or extrinsic muscles are shortened, which causes a pull on the tendons, which, in turn, pulls the according phalanx of the digit. On the contrary, isometric contractions are characterized by a change in the tension of the muscle while its length is constant. This is the case when resistance against the muscle is increased, e.g., when the fingertip is pressed against a rigid surface. This is why force-sensitive input devices, like the IBM TrackPoint (cf. Selker et al., 1991), are also called isometric devices.

Subsequently, we look at psychophysical aspects of force control.

Knowing now how force is applied by the fingers from a physiological perspective, we subsequently look into the psychophysics of the application of force. This includes how much force humans can apply with their finger, how well they can control it, and how their performance is affected to the better or worse by external factors, such as feedback methods, or motion.

2.3 The Psychophysics of Force Input

In this section, we take a closer look at experiments from psychology and HCI that have investigated humans' capabilities and limitations of applying force with hands and fingers.

This section takes a closer look at humans' capabilities and limitations of exerting and controlling force. Evidence is provided through scientific experiments from psychology but also research in HCI has investigated users' strengths and weaknesses when using force as input modality. Besides capturing how much force humans can apply with their fingers and how well they can discriminate different magnitudes of force, we also take a look at the influence of both, internal and external feedback, on human force control. Finally, the choice of fingers and motion affect the performance of force input. Analyzing these influencing factors is indispensable as they will influence the design of force-based interaction techniques.

2.3.1 Maximum Voluntary Contraction (MVC)

The amount of force that a human can

The maximum magnitude of force that humans can apply is referred to as the *Maximum Voluntary Contraction* (MVC). Since

the exertion of force is carried out through the limbs, the MVC is dependent on the flexibility and strength of the corresponding muscles. These parameters relate to the individual development of the muscles, which is why the MVC for each limb varies from human to human [Sosnoff et al., 2005]. The MVC can even be different for the same individual: Jones [1989] showed that the MVC for the elbow flexor muscle varies across the dominant hand (DH) and the non-dominant hand (NDH). The MVC is also susceptible to muscle fatigue. Househam et al. [2004] measured variations in repeated applications of force through hand grip contractions. Subjects were instructed to squeeze a strain gauge at their maximum effort for seven seconds, followed by a 30 s resting phase until the next trial was performed. The data indicated a clear effect of fatigue since subjects' maximum applicable force decreased over trials. Similarly, Jones et al. [1983] identified a fatiguing effect for the elbow flexor. They found that humans can maintain forces at the lower end of their MVC longer compared to forces at the higher end of their MVC.

apply at maximum varies from human to human due to differences in flexibility and strength of muscles. This maximum amount of applicable force is referred to as *Maximum Voluntary Contraction (MVC)*.

Fatigue and also discomfort have been reported by users maintaining high forces during interaction with handheld devices. For example, McLachlan et al. [2013] investigated users' comfort of applying force against the bezel of a touchscreen tablet. While maintaining a force of 6 N was considered comfortable, users remarked that exerting and maintaining a force of 9 N was too uncomfortable for them. For force input via a stylus, this comfort level is even lower: Mizobuchi et al. [2005] found that users felt significant discomfort and fatigue when applying more than 3 N with the stylus on the handheld device.

For interaction with handheld devices, users feel discomfort when applying forces greater than 9 N.

2.3.2 Discrimination of Force Magnitudes

Pang et al. [1991] studied how well humans can discriminate different magnitudes of force. Hereto, subjects were asked to squeeze two movable plates between their thumb and index finger. The plates let the subjects perceive different resistance forces within a range of 2.5–10 N. Over multiple trials, the subjects were given a reference force that they were asked to compare against a second sample force. The experiment revealed a *Just Noticeable Difference* (JND) value of 7%: when comparing two forces, their magnitudes need to differ by at least 7% for the

The *Just Noticeable Difference (JND)* determines the minimum difference between two magnitudes, e.g., of force, such that the human can tell them apart.

human being able to discriminate them. A similar result was also found by Tan et al. [1992]. In addition, Durlach et al. [1989] found that the JND is significantly higher for the NDH than for the DH. Interestingly, the JND decreases when the reference force is maintained for a while.

2.3.3 Controlling Force with Precision

In *force-matching tasks*, subjects are given a force stimulus ans are asked to reproduce it, usually without any external aids, such as visual feedback.

In psychological experiments, the precision with which humans can apply and control force is typically measured though variation in the production and reproduction of different magnitudes of force. Hereto, the subject places her finger on a strain gauge and applies force until the experimenter informs her when the target magnitude is reached. After maintaining that magnitude for a short while, the subject is asked to reproduce it. Throughout these force-matching tasks, no external feedback, such as visible or audible cues, is given. The subject has to rely solely on cutaneous and kinesthetic feedback from skin and muscles, also known as internal feedback.

Humans can reproduce forces best when the magnitudes are within the middle of the individual's MVC range.

Jones [1989] let subjects perform a force-matching task with their arms within a range of 15–85% of their individual MVC (≈169–482 N). The target forces were perceived by one arm first and had to be reproduced using the other arm. Participants were most accurate for reproduction of force magnitudes around 50% of their MVC. For example, they tended to reproduce forces from the lower end of their MVC with significantly less force than required. This was also observed for force exerted using the biceps and triceps muscles [Jones et al., 1982]. Hence, while humans are good at controlling force around the middle of their MVC range, reproduction of force at the extremes of that range is more difficult. This difficulty was also confirmed by Mizobuchi et al. [2005] and G. Ramos et al. [2004] for users performing force input with a stylus on *Personal Digital Assistants (PDAs)* and graphic tablets. For force exerted with the index finger, Slifkin et al. [1999] found that the force variance increases exponentially with the target force.

Using multiple fingers to control low forces decreases a subject's

The precision of controlling force also depends on how many fingers and which fingers are involved. Newell et al. [1994] studied the exertion of force on a dynamometer with the thumb in com-

bination with other fingers of the hand. For low magnitudes of force of a subject's individual MVC range, the addition of fingers decreased subjects' performance, whereas for higher magnitudes, using more fingers increased accuracy of control. The authors concluded that using multiple fingers leads to redundancy and increased sensation of internal feedback. While this is beneficial for high magnitudes, it leads to an overestimation of perceived force for low magnitudes.

accuracy in reproducing these due to redundancy and an increased sensation of internal feedback from skin and muscles.

Wilson et al. [2012] studied the precision of input from each finger exerting force at the bezel of a smartphone held with one hand in portrait orientation. For each digit and different combinations of them, subjects we instructed to exert given target forces. The authors found that participants controlled force most accurately when using the index finger or ring finger, or when applying force with the ring and the little finger together or for the combined exertion via the index and the middle finger. For these fingers and combinations, the error rate was less than 3.2%. For devices held in landscape orientation, Stewart et al. [2010] studied subjects' accuracy of controlling force for different device grips: The authors found that participants were significantly more accurate when controlling force via a two-sided pinch grip with thumb and index finger compared to when using single finger use at the front or against the back of the handheld device.

On handheld devices, using more than two fingers simultaneously for force input decreases users' accuracy.

In summary, force input on handheld devices is most accurate when using not more than two fingers. In any case, if multiple fingers are used from the same hand, they should control the same target force instead of individual magnitudes of force at the same time.

2.3.4 The Influence of Feedback

As for any input technique, feedback can significantly contribute to human precision and accuracy of force input. Basically, feedback can be classified as internal and external feedback. Internal feedback refers to feedback sensed through skin (cutaneous feedback) and muscles (kinesthetic feedback), such as deformation of the fingertip sensed while applying force against a rigid surface. External feedback is a designed feedback given by a system to inform the subject about the intensity of applied force. For ex-

Feedback affects a subject's accuracy of controlling force. Typically, feedback can be classified as *internal* and *external*.

ample, such a system could measure the force and communicate its magnitude visually or via audio feedback to the user.

Internal Feedback

Kinesthetic feedback is more helpful than cutaneous feedback in accurately controlling force.

In general, H. Henningsen et al. [1997] found that humans can utilize information from kinesthetic feedback better than cutaneous feedback for controlling finger force. However, Raj et al. [1985] showed that cutaneous feedback still contributes to the accuracy of discrimination of different magnitudes of force: They anesthetized subjects' fingertips and found that this negatively affected participants' performance. The same effect was observed for anesthetized fingers in weight lifting tasks. Although for such tasks, subjects do not exert force per se, the weights indirectly apply force back onto the finger pads during lifting. Interestingly, while for the thumb and index finger, anesthesia led to an increase in perceived heaviness by 13%–41%, the perceived heaviness for the little finger and the ring finger decreased by 14%–21%.

Compared to pressing against a flat surface, applying force with the finger against a non-planar surface affects humans' reproduction of force negatively.

Henning Henningsen et al. [1995] investigated the contribution of tactile information for both index fingers in a force-matching task. Subjects had to produce low and high intensities with both fingers simultaneously. While one finger pressed against a flat surface, the other one applied force against a conical pad. Independent from the target force, participants always produced less force with the finger on the conical pad. The authors concluded that accurate force production depends on the indentation of the fingertip such that the distortion of the fingertip through the conical pad leads to an increased sensation of force. Hence, for force input on handheld devices, the flat front or back part of the device should be used rather than the curved bezel or sides of the device.

Jones et al. [2006] confirmed that tactile information from the fingertip significantly contributes to the accuracy of exerting force.

Jones et al. [2006] let subjects perform a force-matching task in the range of 2–10 N with and without wearing finger splints on the index finger. Subjects underestimated the perceived force when the finger splints were present, which again confirms that tactile information about the contact surface plays a crucial role in accurate exertion and estimation of force.

External Feedback

External visual feedback significantly increases a subject's accuracy of controlling force.

Jones [2000] investigated the effect of visual external feedback in addition to internal feedback when controlling finger force. Hereto, subjects were instructed to maintain finger forces between 2–6 N and elbow flexion forces in the range of 10–30 N for 120 s. The authors found that when only internal feedback was available, the absolute average error during maintenance was 1 N for finger force control and 4.5 N for elbow flexion force control. With additional visual feedback from a screen, the error decreased significantly to 0.22 N for force controlled via the fingers and to 0.76 N for the elbow flexion forces. These results show that visual feedback is ultimately important for the precision of force control and therefore should also be incorporated in force-based interaction techniques.

Continuous visual feedback leads to a significantly better performance in controlling force compared to providing reduced discrete visual feedback.

A similar conclusion can be drawn from the studies conducted by Mai et al. [1985]. The authors investigated how the amount of visual feedback information conveyed to subjects affects the variance of maintaining force for 20 s. Hereto, the subjects squeezed a force transducer between the index finger and the thumb for force levels lower than 2.5 N. When no visual feedback was given at all, subjects deviated highest from the target force. With little discrete feedback that only indicated whether the participant over- or undershot the current target force, performance slightly improved. However, a significant improvement was found when full continuous visual feedback was given. Similar effects were found by Sosnoff et al. [2005].

Continuous feedback is essential for a good control of force but the effect saturates at a certain point.

Hong et al. [2008] studied the influence of resolution and frequency of visual feedback for force exerted via the index finger. A higher resolution means that the feedback is more continuous, and a higher frequency leads to a lower latency of the feedback updates. When the feedback resolution was low, i.e., rather discrete than continuous, the subjects relied more on their internal feedback and tended to undershoot, while lowering the frequency increased the variance. Furthermore, when visual feedback was overall reduced, subjects showed more individual differences in their performance. The authors concluded that a good continuous feedback is essential for a good control of force. However, the effect saturates at a resolution of more than 128 pixels per N and a frequency above 3.2 Hz.

Using a combination of visual and auditory feedback helps subjects in becoming familiar with controlling force.

Kobayashi et al. [2016] studied the combined use of visual and audible feedback for helping newbies learn how to control force input. Participants were instructed to apply force with their thumb against a foam object. They were most effective in learning to control force input when a combination of both feedback modalities was given rather than a single one. The authors found that using this feedback combination, only five minutes of training is enough to enhance the force reproduction skill for more accurate force input.

Visual feedback overrules tactile feedback.

A. Srinivasan et al. [1996] studied how visual feedback information predominates tactile feedback information perceived from springs. Subjects were instructed to push physical springs of which they could only see a visual representation. The physical and the visual behavior of a spring could either match or mismatch, such that, e.g., the physical spring was much more flexible than the behavior of the visual counterpart. For such mismatches, the study revealed that subjects relied much more on visual feedback, meaning that it overruled the physical sensation. This, again, underlines how important visual feedback is for force control.

Visual feedback also overrules information perceived from auditory or vibrotactile cues while exerting force on a handheld device.

Studies on force input on handheld devices also confirm the importance of visual feedback. Stewart et al. [2010] compared the effect of visual, auditory, and vibrotactile feedback on users' force control performance. Subjects were instructed to exert different levels of force on the handheld device. Since a continuous audio feedback is generally considered irritating [Liao et al., 2006], the handheld device only gave audio feedback to the subject on the transition from one force level to the next one, with three levels in total. Each level was represented by a different music pitch of a plucked string sound. The pitch increased with force level. For vibrotactile feedback, the device produced three discriminable vibration patterns, one for each force level: a short pulse, a series of pulses, and two short pulses. Subjects were 98% accurate when visual feedback was provided, whereas with vibrotactile feedback, participants performed significantly worse with 96–90% accuracy. Subjects could not exceed 83% accuracy when only audio feedback was given.

For accurate menu control via force, visual feedback is more

Wilson et al. [2010] also compared visual vs. auditory feedback for menu control via force on a handheld device. They used spatial audio feedback to give subjects the illusion of having the

menu laid out in front of them with items navigated from left to right. For each force level, the function of the corresponding menu item was spoken out and a unique musical tone was played. While with visual feedback, accuracy rates were 85% on average, providing only audible feedback decreased accuracy down to 74%.

helpful than providing spatial auditory feedback.

In summary, giving visual feedback to the user is essential for a good control of force input via hands and fingers. A continuous visual feedback leads to better performance than a discrete visual feedback, and adding other modalities, such as audible feedback can help the user becoming familiar with force input faster.

In summary, visual feedback should always be provided for force input.

2.3.5 The Influence of Motion

Wilson et al. [2011] investigated the effect of the user siting vs. walking while controlling force on a handheld device. For both conditions, the authors compared two different mechanisms that map the intensity of force to the selection of a menu item: One mechanism was an absolute mapping, whereas the other mechanism was a relative one that mapped force to the velocity of iterating through the menu items. They found that the relative mapping is superior for both, sitting and walking conditions, as subjects were significantly faster and had a significantly higher selection accuracy compared to the absolute mapping. Independent from what mapping was used, the authors found that mobility generally increases error, selection time, and also participants' subjective workload.

Motion affects a subject's accuracy of controlling force input negatively.

A possible explanation for the negative effect of walking could be explained by inadvertent variations in force input. Stewart et al. [2012] investigated the variation of grip force that users inadvertently apply when holding the handheld device and found that it increases with motion. Based on their findings, they recommend a threshold of 0.6 N below which force should be ignored by the system to mitigate inadvertent grip force.

When holding a handheld device, walking leads to inadvertent variations in grip force.

Related to this, Taher et al. [2014] captured users' force profiles during the execution of multi-touch gestures, such as zooming, panning, rotating, tapping, and typing, on smartphones and tablets, while walking. Overall, participants applied more inad-

Compared to small handheld devices, such as smartphones, subjects apply more

inadvertent force on larger devices, such as tablets, while walking.

vertent force on the tablet compared to the smartphone. Rotating and zooming had the lowest magnitude of all gestures, and inadvertent force for typing was lower than for tapping, which caused 0.2–1.6 N of inadvertent force. Also, the index finger (used for all gestures and typing) caused more force than the thumb (additionally used for zooming and rotation). The findings also help defining a threshold that is required to discriminate gesture input from force input. McLachlan et al. [2015] also came to the conclusion that for precise interactions, such as cropping an image or selecting text, the simultaneous use of gestures and force input should be avoided.

In summary, when designing interaction techniques for force input, the designer should consider the affects of fingers being used, feedback being provided, and motion being potentially involved.

In conclusion, when designing force-based interaction techniques for handheld devices, a few things should be kept in mind to improve the user's accuracy in controlling her finger force. First, input at the lower end of the the force range should be ignored, since users inadvertently apply force when grasping the device and especially when being in motion. For this context, a relative mapping rather than an absolute mapping from force to value input should be considered. For high levels of force, input from multiple fingers is better, whereas for low levels, not more than two fingers should be used. Finally, continuous visual feedback must be given to the user. While tactile feedback is important, it is usually overruled by visual feedback, as long as the resolution and the frequency of the visual updates are high enough.

Knowing the design considerations, we now will look into how force can be technically sensed and digitized to utilize it for input on handheld devices.

2.4 Sensing and Digitizing Force

Typically, force is measured indirectly via measurable side effects that correlate with the magnitude of applied force.

In this section, we will discuss different techniques that allow sensing and digitizing force input. Basically all of them determine force indirectly, i.e., they neither measure mass nor its acceleration. Instead, the techniques utilize side effects that occur during the application of force. These measurable side effects correlate with the magnitude of force and hence allow to estimate it indirectly. In HCI, force measured via such indirect measurements is often referred to as *pseudo-force*. Before we present

common pseudo-force techniques that are adequate to determine force via sensors and technology from the handheld device, we will discuss the *Force Sensing Resistor (FSR)*, an electronic component that has been particularly designed to measure force indirectly through changing resistance in an electric circuit. For all techniques, we will highlight their benefits and shortcomings.

2.4.1 Force Sensing Resistors (FSRs)

A *Force Sensing Resistor (FSR)* is a sensor used in an electric circuit that changes resistance depending on how much force is applied to it. Figure 2.5 shows an example of an FSR. An FSR typically consists of three layered components, with a total thickness ranging from 0.2–1.25 mm: The top layer is a substrate layer with screen-printed conductive ink that forms an open circuit, also referred to as active area. The bottom layer substrate holds a conductive foamed film that consists of both, electric and dielectric particles. Between these two layers is a plastic spacer with an air vent. When normal force is applied on the active area, the conductive film is deformed and air is being pushed out through the vent, such that the conductive film comes into contact with the conductive ink from the active area. The more force is applied, the more the two conductive materials come into contact, which lets more current flow through the circuit. Consequently, the electric resistance in the circuit is reduced. Most FSR change their resistance from more than 1 MΩ when no force is applied to a few Ohms when the sensor is driven into saturation. Typical FSR can sense about 1–100 N and have a very quick response time from 1–2 ms. However, the measurements obtained from FSRs are not very reliable: Repeated measurements can be off by more than 10%, especially because the materials wear out easily.

A *Force Sensing Resistor (FSR)* is an electric component that changes the resistance in an electric circuit based on the magnitude of force applied to it.

Rosenberg et al. [2009] built is a high resolution pressure sensing mat that uses *Interpolating Force Sensing Resistance (IFSR)*, a technique that consists of two sandwiched layers of electrodes with a layer of microscopic bumps of FSR ink in between. Due to interpolation, this technology provides a higher grid resolution compared to an equivalent implementation using standard FSRs, i.e., it can discriminate more individual touch locations on the same area. It is a scalable technique that can come in any size, e.g., the size of a credit card or that of a sheet of paper. Fur-

The UnMousePad is a mat that can sense touch and force at any location on the entire surface.

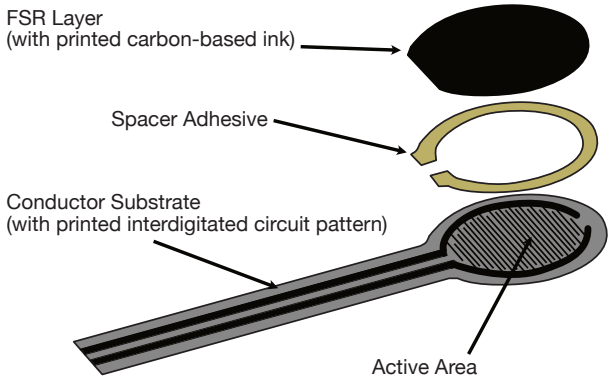


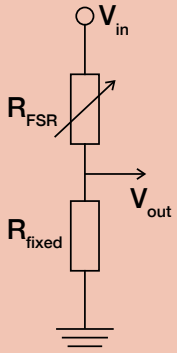
Figure 2.5: Exploded assembly drawing of a *Force Sensing Resistor (FSR)* consisting of three different layers. The more force is applied onto the active area, the more the more current can flow through the printed circuit.

thermore, the UnMousePad can be put behind OLED and e-ink screens to make them touch- and force-sensitive.

→ Knowledge Applied: Digitizing Force Input from a Force Sensing Resistor (FSR) using a Voltage Divider

A basic solution to represent the magnitude of force applied to an FSR as a digital value, e.g., to control an interface, is using a voltage divider circuit that is connected to a microcontroller. A microcontroller, like an Arduino[a], has a set of analog pins that can measure analog voltage, which is then converted into a digital number using an *Analog-to-Digital Converter (ADC)*. Typical ADC have a resolution of 10 bit, i.e., they can represent up to 1,024 different numbers from 0–1,023. The ADC reports a ratiometric value, i.e., it maps a voltage that is as high as the system voltage, usually 5V, to the highest value, i.e., 1,023, and maps lower voltages proportionally using a factor of $\frac{1{,}023}{5V}$. For example, a voltage of 3V is equivalent to $3V \cdot \frac{1{,}023}{5V} = 613.8 \approx 614$.
Recall that the FSR changes its resistance based on how much force is exerted onto the active area. While this cannot feed the analog pin of the microcontroller since it only deals with voltages, we can utilize a specific circuit design, the *Voltage Divider*, with which we can measure

the voltage drop across the FSR based on *Ohm's Law*. The Voltage Divider circuit looks as follows:

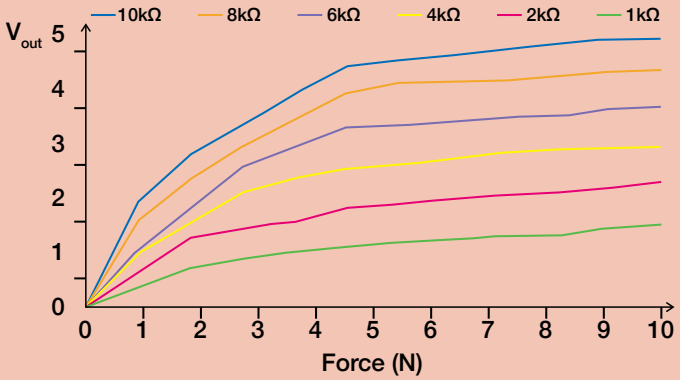


R_{fixed} is a static resistor, which is set to a a relatively high resistance, e.g., 10 kΩ, R_{FSR} represents the FSR as dynamic resistor (also known as potentiometer), V_{in} is the input voltage to the circuit, and V_{out} represents the voltage drop across R_{FSR}, which feeds the analog pin of the microcontroller. According to Ohm's Law, it holds that

$$V_{out} = V_{in} \cdot \frac{R_{fixed}}{R_{fixed} + R_{FSR}} \tag{2.4}$$

When no force is applied, we yield $V_{out} \approx V_{in} = 0V$ since R_{FSR} is relatively high, e.g., 1 MΩ, compared to R_{fixed}, which was set to a static 10 kΩ. The more force is applied onto the active area of the FSR, the smaller R_{FSR} becomes, hence the higher V_{out}, i.e., the voltage drop across F_{FSR}, becomes.

However, while this basic setup is a simple solution to obtain a digital representation of force, the behavior between the applied force and the digital value representation is not linear, i.e., applying double the force does not double the value. Instead, FSRs are typically very sensitive to low magnitudes of force but insensitive towards the end of the sensor force space. The following graph illustrates this behavior for different values for FSR_{fixed}:



The digital value increases rapidly at low force, and continues to grow slowly towards the end of the sensor spectrum. This relation does not match humans' expected behavior, as they associate a linear relationship between force and value [Stewart et al., 2010]. To adjust this, one can either use a function to mathematically linearize the output, or one can add an operational amplifier to the circuit that linearizes the output by hardware. More information on how to wire an operational amplifier to obtain linearized output from an FSR can be found in [Stewart et al., 2010].

[a] https://arduino.cc

2.4.2 Capacitive Force Sensing

Capacitive sensing is used in modern touchscreens to detect finger touches.

Capacitive sensing is typically used in modern touchscreens to detect when and where a finger is touching the screen. However, this technique can also be used to indirectly measure force based on the finger's distance between the touchscreen glass and the underlying sensing technology.

When a user's finger touches the touchscreen digitizer, the electric field between the two capacitor layers located underneath

A capacitive touchscreen consists of a screen, usually a liquid crystal display (LCD) or an organic light emitting diode display (OLED), with a digitizer on top. The digitizer is responsible for detecting touches and consists of multiple thin, transparent layers. Figure 2.6 shows a simplified schematic of the four layers that make up the digitizer [Barrett et al., 2010]. The touch surface sits on top and protects the underlying conductor layers that

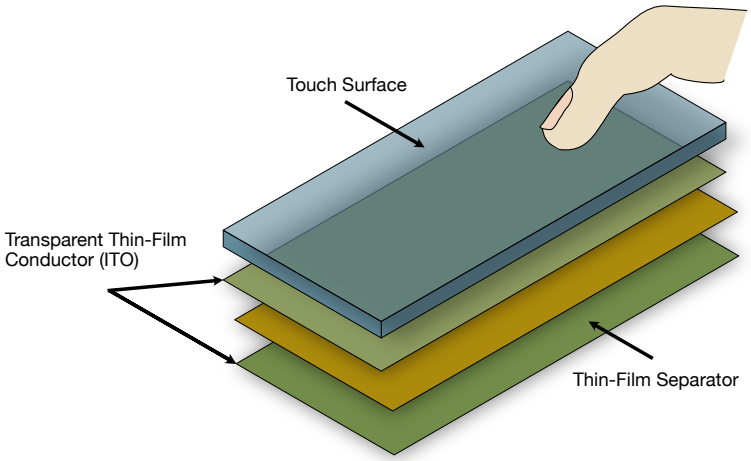


Figure 2.6: Exploded assembly drawing of a capacitative touch digitizer consisting of four layers. Underneath the glass touch surface are two electrically charged *indium thin oxide (ITO)* capacitor plates that are separated by a spacer. When the user touches the surface, the finger splits the electric field between the two ITO layers. The split causes a weakening in the electric field that can be measured and inferred as touch contact between finger and glass surface.

are separated by a thin-film separator. Both conductor layers are made of indium thin oxide (ITO), which is a transparent material that conducts electricity. Technically speaking, these layers act as an array of tiny capacitors whose plates consist of ITO. Each capacitor is charged, resulting in an electrical field between the two plates that is measured by a controller. Now when the user's finger comes into contact with the touch layer, it acts as an additional capacitor plate, since the human body can carry electrical charge. This means that the electrical field is now "split" between two capacitor plates, i.e., between the bottom ITO layer and the top ITO layer, as well as between the bottom ITO layer and the finger. Consequently, the electrical field between the two ITO layers is weakened, which is recognized by the controller and matched as a touch contact.

the display, is weakened and registered as a touch event.

To measure a touch's force intensity using capacitive sensing, as found in Apple's iPhone 6s or newer, a simple trick is from physics is used: The capacitance of a parallel plate capacitor is inversely proportional to the distance between the two plates.

Capacitive force sensing exploits a law from physics: The capacitance of two

parallel capacitor plates is proportional to the distance between the plates. When applying force, the two plates are being pushed closer together.

This means that the closer to the plates are, the larger the capacity for a charge on the plates is. With this in mind, force can be measured indirectly through the distance between the user's finger and the capacitor plate. Hereto, Apple placed a second grid of capacitor plates behind the display and made the touch surface pliable (Fig. 2.7) . The harder the user presses against the screen, the more the pliable surface bends, meaning that the finger comes closer to the capacitor plates behind the screen. The change in capacitance is registered by a dedicated controller, and using physical laws on the relation between distance and capacitance, the force is determined indirectly. Note that detection and localization of the touch are completely separated from measuring force intensity: The digitizer knows *where* a touch is placed, but does not know how much force is applied. The capacitor plates behind the display, however, know *how much* force is applied, but not exactly where. Only the combination of both reveals where and at what intensity the finger is pressing against the touchscreen.

The force-sensitive touchscreen found in Apple's iPhone 6s delivers reliable force readings.

The benefit of this technology is that it is compact and delivers accurate and reliable force measurements since the technique has been specifically designed for measuring force on handheld capacitive touchscreens. However, the technology is complex to implement and cannot be easily added to existing handheld devices since it requires physical modification of the touchscreen.

2.4.3 Piezoelectric Force Sensing

PyzoFlex is a four-layered quasi-transparent foil that senses force via the *piezoelectric effect*.

Rendl et al. [2012] built a hover- and touch-sensitive flexible foil that can also detect force input by utilizing the *piezoelectric effect*. The piezoelectric effect occurs when a piezoelectric material is deformed. Such deformation causes a redistribution of positive and negative charges of piezo crystals. This change in charge can be measured and its magnitude is proportional to the intensity of deformation caused to the piezoelectric material. The force-sensitive PyzoFlex foil consists of a ferromagnetic material that consists of four layers of functional ink. The quasi-transparent ink can be printed on any surface, such as the display of a handheld device. In a technical study, the authors found that technique can continuously sense about 1.25–3 bars of pressure.

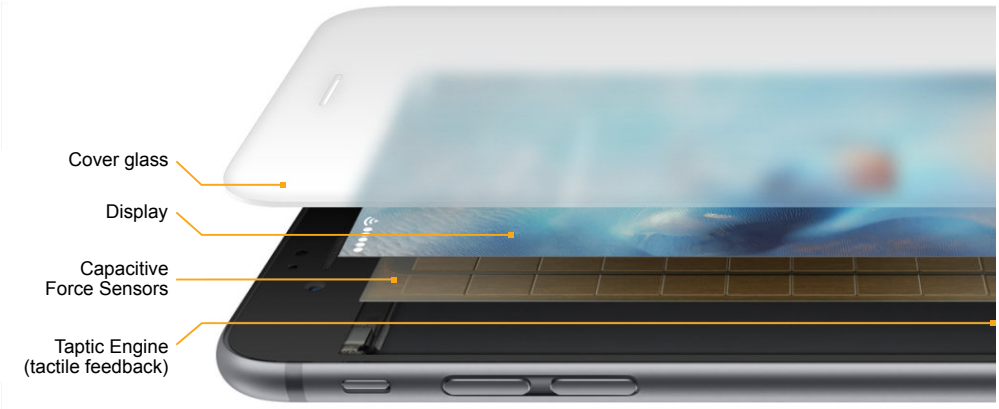


Figure 2.7: Exploded assembly drawing of an Apple iPhone 6s force-sensitive touchscreen. The cover glass is pliable such that the more force the user exerts with her finger, the closer it comes to the grid of capacitive force sensors. This causes a change in capacitance that can be measured and differs based on the distance between the finger and the capacitor, hence, the user's force. Image taken from Apple Inc.

While this technique is flexible and cannot only sense force but also hover and touch input, its resolution is low. Each printed sensor spot has a diameter of 1 cm with a spacing of 1.5 cm to adjacent spots.

PyzoFlex not only senses force input, but also hover- and touch input.

2.4.4 Finger Contact Point

Touch-sensing systems reduce the location of a finger touch to a single point that represents the centroid of the touch ellipsis. Wang et al. [2009a] found that while a finger is landing on a touchscreen, its pad slightly deforms over time. Consequently, the size of the touch ellipsis changes and the centroid moves in the direction of the user's palm. The same effect also holds when applying force with the finger against the touch surface, since also here, the finger pad slightly deforms, such that the touch ellipsis changes its size and its centroid starts moving. Boring et al. [2012] exploited this coherence for estimating force input with the thumb on a handheld capacitive touchscreen that only provides information about the touch centroid. Their technique discriminates two levels of pseudo-force that are utilized to let the user switch between different modes, such as panning and zooming, with a single finger.

The more force a finger applies on a planar surface, the more its touch centroid moves.

Using the finger contact point location as estimator for force is unreliable, since it is ambiguous.

While this such technique adds pseudo-force detection easily to virtually any touch-sensitive surface, it cannot detect many levels of force and might also misinterpret force input: A change in finger contact size and point cannot only be caused by applying force against the touchscreen but also through altering the fingertip angle [Boring et al., 2012]. For example, as the user rocks her thumb on the touch surface, the contact point moves without having applied extra force. However, this slight would be mistakenly interpreted as a change in applied force.

2.4.5 Finger Contact Size

The more force a finger applies against a flat surface, the larger its touch contact size.

Pawluk et al. [1999] showed that contact force against a flat surface can be inferred from the finger's contact size on the surface. In a controlled experiment, participants exerted forces between 0–2 N at angles of 20° and 40° against a flat capacitive sensor array that captured the finger contact size. The authors derived a nonlinear model that approximates a positive correlation between force and the resulting spread of the viscoelastic finger pad, i.e., the finger contact size.

On a vision-based multi-touch table, finger contact size can be used to detect two levels of force input.

Benko et al. [2006] exploited this correlation to detect force input on a vision-based multi-touch table. Their system tracks finger contact optically through infrared light reflected from the fingerpad towards a camera beneath the table. The brighter and the larger the captured spot, the more force has been applied. Using this pseudo-force technique, the system could detect two levels of force to discriminate a hover state from a click state on the touch surface.

As an example for a force-based interaction technique, Baglioni et al. [2011] used finger contact size captured on a handheld capacitive touchscreen to let the user continuously alter the speed of scrolling through content on the handheld screen.

Force detection via finger contact size is finger- and user-specific.

While pseudo-force detection through finger contact size enables touch surfaces to "sense" force across the entire surface without additional hardware modifications, the technique is finger- and user-specific because humans' finger pad size vary [Wang et al., 2009a; Wang et al., 2009b]. For example, one user might have a smaller index finger than another user, which re-

sults in different contact sizes for the same magnitude of contact force. Also, for an individual user, finger pad sizes differ. For example, the thumb creates a much larger contact size than the index finger for the same magnitude of force. This means that a system must be calibrated and trained to discriminate finger pad size differences for providing reliable force estimations. Furthermore, the finger contact size only slightly changes on rigid surfaces like the frontal protection glass of a touchscreen, such that only a few number of pseudo-force levels can be detected [Boring et al., 2012]. Finally, the finger contact size on capacitive touchscreens is often not accessible to the developer. For example, the iOS SDK [Apple Inc., 2019b] only provides the major radius and estimated centroid for each touch ellipsis, which is not sufficient to reconstruct the finger contact size.

2.4.6 Time-Based Force Estimation

Exerting force takes time since the target magnitude must be gradually built up.

A typical assumption is that exerting force takes time. This is because humans cannot instantly apply a certain magnitude of force. Instead, the magnitude must be built up by starting at zero force and then increasing it until the desired magnitude is reached. This takes time: the higher the magnitude, the longer the execution time. In fact, G. Ramos et al. [2004] found that the time required to exert force with a pen on a tablet followed *Fitts' Law* [Fitts, 1954]. Fitts' Law describes that the time to perform linear movements, such as pointing tasks, increases logarithmically with difficulty—the ratio between the distance from the starting location to the target position of the movement and the width, i.e., the circumference of the target position. Typical examples for movements that follow Fitts' Law are pointing tasks using a mouse, touchpad, or touchscreen (e.g., Akamatsu et al. [2000], MacKenzie et al. [1991], Sambrooks et al. [2013], and Soukoreff et al. [2004]). In terms of force input, the distance can be considered the difference in magnitude from zero force to target force, and the width is the tolerance regarding how much the force may deviate from the target magnitude. The correspondence between time and force was also perceived by users in a study from Raisamo [1999]: The author tested interaction with a public information kiosk system where continuous input was once mapped to finger contact time with the touchscreen and once to true force readings from the touchscreen. The sub-

jects reported that interaction via the time-based technique felt almost identical to the true force-based technique.

`Pressure.js` is a library that uses touch contact time to simulate force input.

One example using time to estimate force is `Pressure.js`[1], a JavaScript library to handle force input in web-based applications. If the input device does not natively support force input, like a computer mouse, the library simulates the force based on how long finger contact is maintained, e.g., with the mouse button.

Using time as an estimation for force slows down experienced users.

However, time is generally a bad estimate for force input since the execution time for force might differ across users and interfaces. Imagine a UI with a binary mode switch that is controlled by low vs. high force. This is fairly easy and quick to control since the user can just apply a little force vs. significantly more force to alter the mode. If, however, more precise control is needed, e.g., when force input is mapped to a menu with multiple items to select from, then especially unexperienced users will need more time to build up force compared to experienced users. Referring back to Fitts' Law, this corresponds to an increased difficulty since the width, or tolerance, respectively, decreases with the number of menu items. Furthermore, time-based pseudo-force slows down experienced users since they must wait until the corresponding time estimating a certain force magnitude has elapsed although they would have reached this force level much earlier, e.g., with an FSR. To address this problem, Arif et al. [2013] combined time-based pseudo-force with pseudo-force based on the movement of the touch contact point. Whatever event occurs earlier triggers the next force level. This way, the authors could reliably discriminate two levels of force without the user having to perform one tap slower than the other. In an improved version, Arif et al. [2014] were able to discriminate up to seven force levels using this combined technique.

2.4.7 Vibration-Based Force Estimation

Force input dampens the signal of a constantly pulsing vibration motor, as

Goel et al. [2012] utilized the built-in vibration motor and gyroscope from a smartphone to detect pseudo-force. Hereto, the authors pulsed the vibration motor while using the gyroscope as feedback channel. When the user's finger touches the screen,

[1] https://pressurejs.com

it partially dampens the vibration, which affects the signal captured by the gyroscope: the harder the user presses against the touchscreen, the more vibration is absorbed, and the weaker the signal captured by the gyroscope is. Passing that signal through high- and low-pass filters to feed a decision tree algorithm, the authors were able to discriminate a light press from a hard press with 97% accuracy.

found in smartphones. The more force is applied, the more the signal is dampened.

Hwang et al. [2013] used a similar approach but captured the vibration feedback using the built-in accelerometer from a smartphone. They could discriminate four levels of pseudo-force with 92% accuracy.

Heo et al. [2011b] dispensed the pulsing of the vibration motor and only captured acceleration along the axis normal to a handheld touchscreen. This is the axis along which the device slightly moves when the user touches the screen. Depending on how strongly the user taps, different acceleration patterns occur. Using this pseudo-force effect, the authors successfully discriminated gentle tapping from strong tapping above 90% of all times.

Even the accelerometer of a smartphone can be repurposed to discriminate at least two levels of force input.

While these techniques utilize built-in technology to add pseudo-force to smartphones, the permanent pulsing of the vibration motor not only makes constant noise but also consumes significant battery power. While this is not an issue for Heo et al.'s implementation, their technique needs to be calibrated carefully based on the contexts the device is used in, such as standing, walking, or having the device resting on a table, since these contexts affect the magnitude of acceleration differently.

Pulsing the vibration motor is noisy and consumes significant battery power.

2.4.8 Acoustics-Based Force Estimation

Hwang et al. [2012] used the built-in speakers and microphone to detect pseudo-force on a smartphone. The speakers emit ultrasonic sound whose feedback is recorded by the microphone. When the user puts her finger on the microphone, the feedback is interfered. The stronger the user presses her finger against the microphone opening, the more sound is blocked. Passing the recorded signals on to a *Fast Fourier Transformation* algorithm, the authors were able to classify four different force levels with 94% accuracy.

A combination of speaker and microphone, as typically found in smartphones, can also be used to detect force input.

With pseudo-force technique, force can only be applied at the microphone opening.

While speakers and a microphone are readily availably on smartphones, the technique has a major shortcoming: force input can only be performed at a single dedicated location: the microphone opening. Also, ambient noise can interfere with the detection.

2.4.9 Pressure-Based Force Estimation

When pressing against a smartphone touchscreen, the pressure of the air inside the device increases. This pressure change can then be captured by the device's internal barometer sensor.

Another way to sense force input on handheld devices is by repurposing the device's internal barometer sensor. When the user presses with her finger against the touchscreen, the display slightly flexes inward. If the device is airtight, the air pressure increases since the internal air cannot escape. This change in pressure is detected by the internal barometer. Takada et al. [2019] exploited this effect and studied the correlation between gravitational force and the barometer readings using different weights placed on the touchscreen of a smartphone and a smartwatch. Based on this data, the author's technique can detect two to six levels of force with 96% accuracy.

Quinn [2019] also exploited this effect to add force to a variety of smartphones. They built a linear calibration model that relates pressure to force and that can be tuned to individual users and devices. The technique can continuously track force input within a range of 1–5 N with a sensitivity of less than 1 N.

Due to the low sampling rate of the barometric sensor, too slow and too quick changes of pressure, and thus force, cannot be detected.

While this technique can be used to add force input to handheld devices without hardware modifications, it has two major limitations: First, due to the low sampling rate of the barometer and equalization of pressure, the technique cannot detect too quick or too slow changes in force, i.e., pressure. Second, the technology is not compatible with multi-touch. Hence, the model always assumes that the changes in pressure are only caused by a single finger.

2.4.10 Light-Based Force Estimation

Force can also be detected visually.

Watanabe et al. [2012] used light transmitted through the user's finger to detect pseudo-force on a handheld capacitive touchscreen. When light, e.g., the touchscreen's display backlight,

is pointed from beneath the user's finger, the red light component passes through flesh and fingernail. The passed-through light intensity increases nonlinearly with the magnitude of force when the finger presses against the light source. When the user touches the screen, the display shows a bright white spot beneath the user's finger, while the rest of the display is darkened. A camera with a filter attached on top of the fingernail captures the light transmitted through the finger. While this is an interesting approach, it is, however, not practical, especially for the mobile context since the user is required to wear a camera on her finger. Furthermore, the display beneath the touchscreen cannot be used for showing the graphical UI and content anymore.

Having understood what force is, how well humans can control it, what factors influence force control to the worse and better, and how it can be sensed and digitized, we will now look into existing interaction techniques that use force as input method. In particular, we will look at force-based interaction techniques for the use in stationary environments, such as the desktop, and for mobile, handheld use.

Subsequently, we will present existing force-based interaction techniques. We will look at two device classes: stationary and handheld devices.

3

Related Work Force-Based Interaction

This chapter presents a variety of interaction techniques from HCI literature that use force as input method. We classify the techniques regarding their context of use, i.e., either in a stationary or a mobile environment and discuss its application of use. To begin with, we present a basic interaction technique, called *Pressure-Based Linear Targeting (PBLT)*, that is typically found in both, the stationary and mobile context. Furthermore, we discuss important design parameters that fundamentally characterize each force-based interaction technique. Based on these design parameters, we close the chapter with a tabular overview of almost 80 research papers that present studies and interaction techniques about force input. The overview aims at simplifying comparisons across the related work. hat lay the foundation for each force-based interaction technique.

This chapter presents related work about force-based interaction techniques. In particular, we focus at such interaction techniques used in stationary and mobile contexts.

3.1 Pressure-Based Linear Targeting— A Basic Force Input Technique

A basic input technique based on force input is *Pressure-Based Linear Targeting (PBLT)*. In summary, PBLT lets a user select a discrete item or value from a one-dimensional space, like a menu or list, based on how much force she applies. Subsequently, we will explain how PBLT is designed. For this, we assume the fol-

A basic force-based interaction technique is *Pressure-Based Linear Targeting (PBLT)* that maps the

continuous, digitized force values to equally-sized discrete bins. This technique is typically used to control linear menus via force.

lowing characteristics from the force sensor, e.g., an FSR: The sensor continuously delivers digitized force readings with a resolution of 10 bit, i.e., values from 0 to 1,023, and shows a linear behavior, which means that a doubling in force results in a doubling of the mapped digital value. Furthermore, we assume the sensor to capture forces from 0 to 4 N until it is driven into saturation. This sensing range is also know as the pressure space of a force sensor. Based on our naming convention (cf. Force vs. Pressure), we will refer to this as force space throughout the thesis.

PBLT uses an absolute mapping: it maps the entire force space to the entire value space.

PBLT uses an absolute mapping from force space to value space. The value space is a one-dimensional space of discrete values or items that the user wants to select from. An apt example is a list with seven items, say, weekdays, that are sorted from Monday to Sunday. Since the force sensor usually delivers more values than both, list items to select from and what the human can reliably control, the force space is binned into equally-sized levels, each of which is then naturally mapped to a different item form the list. In our case, the 1,024 values are split into seven bins, which yields about 146 force values per bin, such that we map force values 0–145 to the first item, i.e., Monday, force values 146–291 to Tuesday, and so on. Figure 3.1 shows the complete mapping. This mapping is a *natural mapping* [Norman, 2002]: The more force is applied, the further the selection moves towards the end of the week. A visual highlight typically feedbacks to the user which item is currently matching the exerted force. The blue color in Figure 3.1 illustrates this for both, discrete and continuous visual feedback.

3.1.1 Hysteresis

A *hysteresis* prevents fluctuation of preselecting two adjacent PBLT levels when the user's force is jittering.

In order to prevent jumping back and forth between two adjacent levels when the user's magnitude of force is close to the border of the two items, a *hysteresis* is used. A hysteresis defines a tolerance range around the exact border of the two adjacent levels to prevent fluctuation: When the user reduces force, the lower level is not preselected until the lower bound of the tolerance range has been undershot. Likewise, the upper level is not preselected until the upper bound of the hysteresis has been crossed. Hence, the hysteresis stretches the exact border between two ad-

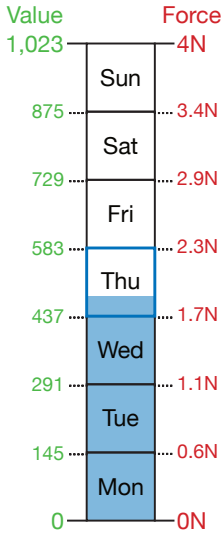


Figure 3.1: Binning example for selecting a weekday via *Pressure-Based Linear Targeting (PBLT)*. The continuous sensor force space of 4 N is linearly mapped to digital values from 0–1,023 (10 bit). This value space is equidistantly split into seven bins, with each bin representing a different weekday. To select a weekday via PBLT, e.g., Thursday, the user must exert force in the range of (1.7 N; 2.3 N], which is equivalent to a digitized value in the range of [438; 583]. The blue color indicates which item (here: Thursday) is currently selected.

jacent levels by adding some buffer space and takes the direction of force, i.e, whether is is currently being increased or decreased into account.

3.1.2 Selection and Confirmation

Force input is *transient*, i.e., the magnitude can only be maintained by the user actively maintaining the force.

In general, force input is transient, i.e., the magnitude can only be maintained by the user actively maintaining the force. This is different, e.g., from mouse input in desktop systems. Here, the user can place the mouse cursor over a menu item and remove the hand from the input device without the cursor being reset to the position where the user started from. Furthermore, to select an item from the menu via force, the user must pass through all items listed before the target item. This is because the target force cannot be achieved instantly since force is based on accel-

eration that needs to be built up, i.e., grow over time. Likewise, while releasing force, all items listed before the current item are traversed in reversed order until force is completely zero. This mechanism is also known as *natural inverse* [Ghazali et al., 2005]. Hence, a direct selection, like tapping on an item on a touchscreen, is not possible via force input.

Selection of an item via force input is ambiguous. To solve this problem, dedicated *confirmation techniques* have been designed.

Since an item can only be navigated to indirectly, i.e., by passing intermediate items, this makes actual selection of an item ambiguous: How does the system know that the user finally reached the target item without mistakenly interpreting an intermediately passed item as the target item? To solve this selection problem, different *confirmation techniques* have been designed. Subsequently, we present the three most common confirmation techniques: *Thresholding, Dwell Time,* and *Quick Release*.

Thresholding

Thresholding sets an item as preselected once its lowest force value has been crossed.

Thresholding sets an item as preselected once its lowest force value has been crossed. The preselected item is updated as soon as the next highest force boundary has been passed. This means that thresholding only allows to change the preselection in one direction, namely only upwards along the force space. Consequently, once the preselection moves from one item to the next, the previously preselected item cannot be reached again unless the entire selection process is restarted. When force is removed from the sensor, the currently preselected item is confirmed. Hence, force and value are decoupled as soon as the force drops below the last crossed boundary.

Thresholding only allows for navigating in one direction along the selectable items. This means that accidental overshoots cannot be corrected unless the entire selection process is restarted.

A benefit of this confirmation technique is that the user does not need to maintain exact force to stay within an item. Furthermore, from a technical point of view, the confirmation procedure is clearly defined and therefore easy to implement. However, the user cannot easily correct for accidental overshoots since Thresholding only allows for a unidirectional navigation. If the user applied too much force, she will have preselected an unwanted item, unless the menu provides a cancellation option that is, e.g., mapped to the upper force space. In any case, correcting for overshoots takes significant time.

Dwell Time

Dwell Time requires the user to maintain force within the boundaries of the target item for about 1 s.

Unlike Thresholding, *Dwell Time* features bidirectional navigation and thus enables the user to easily correct for over- and undershoots. To finally select an item, the user maintains force within the boundaries of the corresponding level for a specific time, called *dwell time*. Most interaction techniques using this selection mechanism set the dwell time to be about 1 s (e.g., Brewster et al. [2009], Cechanowicz et al. [2007], Shi et al. [2008], Stewart et al., 2010, and Wilson et al. [2011]). As soon as the dwell time has expired, force and value are decoupled such that the user can lift off her finger from the sensor without the force drop affecting the selection.

Confirmation via Dwell Time leads to high success rates, but it artificially slows down the selection process.

Like Thresholding, the Dwell Time confirmation technique is clearly defined and therefore easy to implement: When the user exerts force, a countdown timer is started for the specified dwell time and the current item is preselected. If the user changes force significantly such that the preselection changes to a different item, the timer is reset. Otherwise, if the timer expires, the preselected item is confirmed. Studies around interaction techniques that use Dwell Time for confirmation of force input have shown that users achieve high success rates of about 97% in PBLT tasks (e.g., Brewster et al. [2009], G. Ramos et al. [2004], Stewart et al. [2010], and Wilson et al. [2011]). A typical drawback of this confirmation technique is the increased selection time: Compared to Thresholding, Dwell Time will take longer since the user has to wait for until the dwell time is exceeded before the selection is finally confirmed. Another disadvantage is that Dwell Time does not support exploration very well: Imagine force is used to navigate a menu with each menu item representing a different option or mode in the UI. If the user wanted to explore all options and see their effect, she could simply press to navigate and display the effect and inspect it. However, if that inspection takes longer than the dwell time, this option will then be automatically selected and exit the menu.

Quick Release

To confirm force input via *Quick Release*, the

Like Dwell Time, *Quick Release* features bidirectional navigation and makes it easy for users to correct for over- and undershoots.

user simply quickly lifts her finger off the force sensor. However, reliably interpreting the user's intended selection is difficult.

To select an item, the user applies force until the intended item is preselected and then she quickly lifts her finger off the sensor. Compared to Dwell Time, Quick Release makes the confirmation of force input very efficient because the user does not need to wait for the dwell time to expire (Fig. 3.2). Furthermore, Quick Release benefits exploration of modes: even when the user lingers longer on an item, it will not confirm selection automatically, as would happed for Dwell Time. However, compared to Dwell Time and Thresholding, reliably detecting the Quick Release event is challenging: *"[...] Designing an accurate Quick Release mechanism is troublesome because it is difficult to identify a common and clear sensor behaviour from which user intent can be unambiguously retrieved."* [Wilson et al., 2010]. In fact, error rates range between 5–40% (e.g., Brewster et al. [2009], G. Ramos et al., 2004, Stewart et al., 2010, and Wilson et al., 2011).

The force pattern used to detect a Quick Release event is ambiguous.

A quick drop in force can be caused by two different events: On the one hand, the user might have quickly lifted the finger off the sensor to confirm input. On the other hand, the user might have reduced force quickly to correct for an overshoot. In any case, the force sensor registers a quick drop in force. A reliable statement for the Quick Release event can only be made when the sensor measures zero force *afterwards*. Only then, the user must have completely lifted her finger off the sensor, which is not the case during the correction process for an overshoot.

More detailed information about Quick Release and how we challenged the low reliability of it is presented in Chapter 5.

3.1.3 Transfer Function

The transfer function determines how exactly each magnitude of force is mapped to a digital value.

So far, we have assumed that the mapping from force to digitized values is linear, i.e., doubling the force results in a doubling of the digitized value. However, this is must not necessarily be the case. Hereto, the so-called *transfer function* determines how exactly force is mapped to the digitized values. While for PBLT, linear transfer functions are recommended [Stewart et al., 2010], researchers have also experimented with alternatives, such as quadratic- [Cechanowicz et al., 2007], logarithmic- [Corsten et al., 2019], fisheye- [Shi et al., 2008], or sigmoid functions [Ren et al., 2007]. The latter has an "S"-shaped curve, which means the

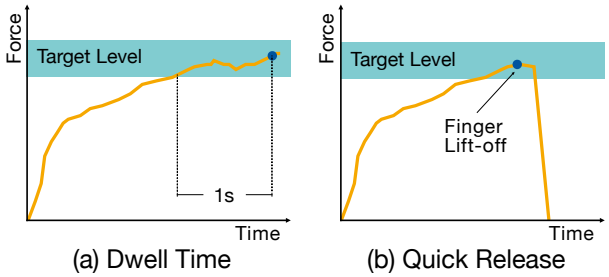


Figure 3.2: Force curves for confirmation of force input via (a) *Dwell Time* and (b) *Quick Release*. The user wants to select a target force level via PBLT. To confirm her input via the Dwell Time mechanism, she needs to maintain the force for about 1 s within the target level. To confirm her input via the Quick Release mechanism, the user quickly lifts her finger off the force sensor, once the target level is reached. Consequently the force drops quickly to zero. Selecting values via Quick Release is faster since the user does not need to wait for 1 s until input is confirmed. Graphic (slightly modified) taken from G. Ramos et al. [2004].

lower end and the higher end of the force space have a plateau, meaning that the exerting of force does not change the digitized value. However, a plateau at the higher end of the force space is typically given automatically, since when the force sensor is driven into saturation, any any increase in the magnitude of applied force cannot be registered by the sensor, and hence, will not change the digitized value.

In general, most transfer functions have in common that an increase in force also results in an increase in the digitized value—this is considered a natural mapping [Norman, 2002].

3.1.4 Control Mechanisms

A more basic design decision is the control mechanism that determines how force controls navigation through the values and preselection of them. Two common control mechanisms are *Position-Based Control* and *Rate-Based Control*.

Position-Based Control

With *Position-Based Control*, the same force value always maps to the same position in the force space.

Position-Based Control, or *Positional Control* establishes an *absolute* mapping from the (binned) force space to the value range, with each level being associated with a different value. This is the mapping used in PBLT. It is called *positional* because it has a consistent force-to-value mapping: The same force always maps to the same value, or position, in the force space.

Position-Based Control allows the user to quickly correct for over- and undershoots, but using this mechanism, it is not feasible to control more than six items reliably via force.

Next to its consistent mapping, positional control has the benefit that the user can quickly correct for over- and undershoots. In case of an overshoot, the user simply reduces force, whereas in case of an undershoot, she can simply increase the applied force. However, a significant drawback of Position-Based Control is that humans can typically not control more than six items reliably (e.g., Mizobuchi et al. [2005] and G. Ramos et al. [2004]). Already at ten items, success rate drop below 90% (e.g., Corsten et al. [2017a], [McLachlan et al., 2014], and Wilson et al. [2010]).

To control more than six items via force input, *Rate-Based Control* should be used.

Of course, the limit of controllable items also depends on the force space used. However, as described in Section 2.3.1, users start to feel discomfort and fatigue when exerting forces more than 9 N with their fingers [McLachlan et al., 2013]. More information about force-based techniques using Positional Control is presented in Section 7.2.1. To control a larger number of items via force, a different control mechanism is recommended: *Rate-Based Control*.

Rate-Based Control

Rate-Based Control maps force to the velocity of iterating through the list of selectable items.

Rate-Based Control, also often referred to as *Velocity-Based Control* maps the (binned) force to the speed at which the items are iterated. Hence, Rate-Based Control uses a *relative* force-to-value mapping. Imagine setting a countdown timer to a value between 1 and 60 minutes. Initially the timer is set to zero minutes. Since the amount of items, 60, is way larger than what can be reliably controlled using PBLT, force input via Rate-Based Control could be used: If no force or only very low force is applied (level 1), the speed is zero, i.e., the countdown timer stays at zero minutes. When the user applies a little more force, the speed could be set to 1 item per second, i.e., as long as the user maintains

that force, the timer value increases by 1 minute every second. If the user applies significantly more force (level 2), the iteration speed could be set to 2 items per second, i.e., every second, the timer value increases by two minutes, which is equivalent to incrementing the timer value every 500 ms. Finally, at level 10, the speed could be set to 20 items per second, to significantly speed up iteration through the value range.

Instead of binning the digitized values, Rate-Based Control typically uses a continuous transmission between force input and value output.

Since it is not important that the user exactly controls or selects a particular velocity, a binning of force to velocity is not compulsory. In fact, rate-based techniques rather use continuous force input to ensure a fluent change in velocity. To speed up the velocity at higher magnitudes of force, exponential transfer functions are often used (e.g., Antoine et al. [2017]). This means that, unlike PBLT's binning, event the slightest change in force will directly affect the velocity[1].

By default, Rate-Based control only supports unidirectional value navigation.

By design, Rate-Based Control requires no particular confirmation technique. When no force is applied, the velocity is zero, i.e., iteration through the values just stops. A significant drawback of this control mechanism, however, is that it only supports unidirectional navigation: The user can only increment the value (i.e., the minutes in the countdown timer example). To decrease the current value, e.g., because the user accidentally overshot the target value, a second force sensor could be used. Alternatively, after having reached the maximum value, the iteration through the values could start again at the lowest value. This wrap-around technique, however, is tedious and time-consuming. More detailed information about interaction techniques using Rate-Based Control and how to tackle bidirectional navigation is presented in Section 7.2.2.

Next, we will take a look at HCI research about input devices and interaction techniques that utilize force input.

Having explained basic design considerations for force input, we will now take a look at more specific interaction techniques that have been researched in HCI. This will help understanding the benefits of using force input throughout the interaction. Since the first force-based input devices and interaction techniques were designed for the use in stationary environments, such as the desktop environment, we will first highlight papers from this area. Subsequently, we will focus at force input for handheld device use.

[1] Of course, this also depends on the transfer function used.

3.2 Force-Based Interaction Techniques

In this section, we present on overview of relevant interaction techniques that use force input on handheld devices. However, since the first force-based interaction techniques and input devices we designed for the stationary context, we will highlight these works first. As we will see, many interaction principles of these works are also found in the handheld techniques.

3.2.1 Stationary Force Input Techniques

This section highlights research about input devices that use force to augment stationary desktop interaction. The force-based interaction techniques are presented by device category: mice, joysticks, styli, and touch pads. As an excursus, the section closes with force input in a similar, yet different stationary environment: car cockpits.

Computer Mice

HCI researchers have experimented with force input on computer mice.

The computer mouse is the ultimate pointing device to interact with desktop user interfaces. Originally invented in the 1960s by Engelbart and English [English et al., 1967], today's mice are usually equipped with two or more physical buttons and a scroll wheel for point-and-click interaction with *Windows, Icons, Menus, and Pointers (WIMP)* interfaces and to easily scroll through documents, such as web pages. To make desktop interaction even more efficient and expressive, HCI researchers have experimented with adding force input to computer mice.

Omata et al. [2007] added a force sensor on top of the left mouse button to let the user control the level of detail required in navigation tasks.

Omata et al. [2007] added a force-sensing physical button on top of the left mouse button to make navigation tasks more efficient: The authors mapped the button force to the level of detail shown in media, such as maps and websites. The more force is applied, the more details, like minor streets on the map, are displayed. Clicking softly on a link on a web page opens up the associated web page only showing its text, whereas a strong press displays the full content, including style layout and images. However, in

a controlled experiment subjects were still faster using a conventional mouse with and without a scroll wheel for navigating such tasks.

By adding two FSR to a mouse, Cechanowicz et al. [2007] enabled two-finger navigation of menus with up to 64 items.

Cechanowicz et al. [2007] added two FSRs to the sides of a mouse chassis to enable the thumb and the middle finger of the steering hand to navigate a menu with up to 64 items. This is done in a two-step approach, called *Tap-and-Refine*: First the user taps multiple times with the thumb on the FSR to specify a coarse level and then she applies force to refine, i.e., doing a fine-grained selection within that level via PBLT. To confirm, the user clicks with the index finger on the left mouse button. For efficiency, the middle finger and the second FSR are used to navigate the menu in reversed order.

Shi et al. [2008] designed a fisheye transfer function for force input via the mouse that that dynamically increases the force boundaries around the level of interest.

Instead of using two FSRs and a two-step navigation approach, Shi et al. [2008] used a single FSR. The authors investigated different discretization functions to both increase the number of controllable items in PBLT tasks and to reduce the time to select an item via the mouse. Among other discretization function designs and a baseline condition that equidistantly splits the force space into the desired number of discrete levels, a fisheye mapping that dynamically increases the force tolerance around the level of interest, was tested. Compared to the baseline condition, Shi et al. found that the fisheye mapping improved users' force control performance without compromising speed when navigating more than 10 levels.

Using rate-based control, Shi et al. [2009] designed a technique that lets users rotate objects by pressing the mouse button.

In a follow-up work, Shi et al. [2009] used the same hardware to investigate force mapping functions that enable dragging of virtual objects and simultaneous rotation of them. They tested a rate-based mapping that increases angular speed based on the intensity of force against absolute force-to-angle mappings and classic dragging and rotation via the mouse. Users were significantly faster with the rate-based force mapping and could control it better than the absolute force-based alternatives.

A force-sensitive mousepad by Kuribara et al. [2015] lets users control the stack of overlapping windows in a desktop

Instead of adding force sensors to the mouse, Kuribara et al. [2015] added an 8×8 FSR matrix beneath a malleable mousepad. The device senses when the user pushes the mouse into the mousepad or when she tilts the mouse by pushing its end into the pad. These gestures are coupled to navigate a stack of overlapping desktop windows: Tilting lets the user quickly browse

environment by pushing the mouse into the pad.

stepwise through the window stack and pushing immediately pops the background window to the front.

Touchscreens, Tablets, and Pads

Touchscreens, tablets, and pads all sense a user's finger touch input.

Touchscreens combine touch input via a digitizer and output via a screen in one device. Since the digitizer and the screen share the same dimensions and are co-located, touchscreens provide the illusion of direct manipulation [Shneiderman, 1997]: the user directly interacts with the virtual object underneath her finger. By contrast, a tablet—not to be mistaken with modern tablet computers, such as iPad—is merely a digitizer that controls a UI displayed on an external screen via indirect but usually absolute touch input. Trackpads are similar, but are significantly smaller than tablets and use a relative mapping to control a cursor on an external screen. All three input devices are also used for input via gestures, e.g., swipes.

The first force-sensitive touchscreen was presented in the late 1970s and could sense eight different properties for a single touch point.

In 1977, Herot et al. [1978] presented the first force-sensitive touchscreen that senses eight properties for a single touch: position in x and y, tangential force, i.e., shear, in x and y, normal force, and torque around the x-, y-, and z-axis. The system consists of a CRT screen with a touch digitizer based on *acoustic wave sensing* (Fig. 3.3): Two piezoelectric transducers mounted to the horizontal and vertical axis of the CRT induce acoustic waves into the screen glass that are reflected back upon finger contact, such that the touch location can be inferred. Between the glass and the digitizer, nine strain gauges are mounted to detect shear, force, and torque with a resolution of 12 bit per dimension. Herot et al. showed how these dimensions could be used for drawing vectors and paths with single finger contact: To orient the vector, the user shears her finger in the corresponding direction, whereas normal force determines the vector length. Similarly, shearing determines the current orientation of the path tail, whereas normal force controls the drawing speed. This metaphor also allows to displace virtual objects. Using torque around the z-axis, the user can rotate a virtual knob using a single finger.

Minsky [1984] demonstrated a

Minsky [1984] used a similar technique as Herot et al. [1978] to sense force input on a CRT with just four strain gauges mounted

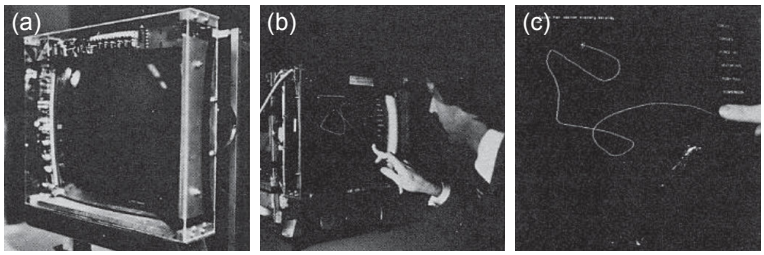


Figure 3.3: Force-sensitive CRT touchscreen by Herot et al., 1978. (a) Side view of the touchscreen that senses touch, torque, tangential- and normal force via nine strain gauges. (b,c) The user is drawing a line using a single finger: pressing against the screen determines the length of the path, whereas shear force controls the orientation of the line while it is being drawn on the CRT screen. Image (slightly modified) taken from Herot et al., 1978.

behind the corners of the screen glass. Her system demonstrated several interactions that utilized force input, e.g., to enable the user to increase the blob size for finger painting by pressing harder against the screen. Force is also used to discriminate engagement with a virtual object or button from displacing it: A hard press triggers the button's event, whereas a light press activates a dragging mode that allows the user to let the object follow her finger. *Minsky's Force Screen* is also the first force-sensitive input device plotted at Card et al.'s *Design Space of Input Devices* [S. K. Card et al., 1990].

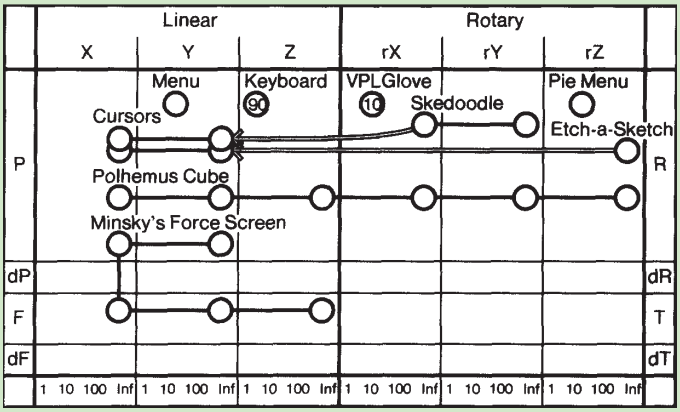
force-sensitive screen for a finger painting application that enables users to control the blob size via force.

→ Excursus: The Design Space of Input Devices

S. K. Card et al. [1990]'s *Design Space of Input Devices* provides a seminal overview of input devices for classifying and comparing them. Since the design space also allows to predict (currently) inexistent input devices, it is more powerful than toolkits or taxonomies.

The design space consists of different dimensions along which the input devices are classified. Each dimension specifies what physical manipulations from the user are sensed by the device. Furthermore, the design space classifies the transmission of the input as absolute

or relative and it also captures the resolution of that input, i.e., the number of different input states that can be sensed in that dimension. The following figure taken from Mackinlay et al. [1990] shows the layout of the design space and plots a few exemplary input devices.



An input device is depicted as a set of connected dots. Each dot represents a manipulation characteristic that is sensed by the device. A manipulation is classified as pointing (P), rotation (R), force (F), or torque (T). Relative input of these manipulations is defined by a 'd' (for 'delta'). The further a dot is positioned to the right, the more continuous the sensing of this manipulation is. The design space also determines along which dimension the manipulation can be captured (i.e., in x-, y-, and z).

As an example, we take a look at *Minsky's Force Screen* [Minsky, 1984]. The input device consists of five connected dots. Two of them (P–X and P–Y) indicate that the screen can sense absolute pointing input in x and y—the user's touch input on the screen. The remaining three points (F–X, F–Y, and F–Z) indicate that Minsky's Force Screen also senses force input along these three dimensions, i.e., shear force in x and y, and normal force along the z dimension. All points are located to the right within each cell (towards '*Inf*inite'), meaning that for along each dimension input is sensed at fine granularity.

In 1981, Sasaki et al. [1981] presented the first capacitive touch tablet. The tablet measures the capacitance at the finger's contact point and serves as input device to a co-located screen. Furthermore, the tablet can determine the finger's applied force indirectly through the finger's contact size on the tablet surface: with force, the contact size grows as does the capacitance. As an example, the authors presented a synthesizer application for generating FM sounds via touch and force: the finger location determines timbre and pitch, whereas the force intensity controls the volume of the sound.

Sasaki et al. [1981] built a capacitive touch tablet that estimates the user's force input via contact size between the finger and the device.

Lee et al. [1985] added multi-touch support to the capacitive tablet, and Buxton et al. [1985] presented further interactions utilizing the touches' z-dimension on this device: While for touch input, a system cannot discriminate between hover and click as known from mouse-based interactions, force can actually help discriminating these two states: Dragging the finger with light force across virtual objects on the touch surface is equal to a hover mode that then transitions into a "click mode" as soon as the user increases her force. Buxton et al. also enhanced virtual painting on the capacitive tablet: a light press positions and moves the drawing cursor, whereas pressing firmly and moving the finger across the tablet actually draws the path. Utilizing continuous force, the user can alter the stroke width while drawing. However, Buxton et al. found the increased friction between finger and surface to be problematic for such drawing tasks.

Buxton et al. [1985] utilized force to discriminate between a hover- and a click state when using a tablet to control virtual buttons. A light press triggers the hover state, whereas a firm press actually triggers the button event.

Blaskó et al. [2004] investigated one-handed force input techniques for a capacitive trackpad. They divided the trackpad surface into four same-sized strips. Similar to a chording keyboard, each strip is controlled by one finger of the hand except the thumb. Within each strip, a finger can tap or drag vertically to control virtual widgets, such as a set of buttons or a spring-loaded wheel. By applying force, the user pops through different widget types within each strip. Pressing two adjacent strips simultaneously gives access to a third strip lying virtually in between the two physical ones.

Blaskó et al. [2004] used a force-sensitive trackpad to enable the user to navigate through different layers of overlapping widgets that are used for scrolling.

Rekimoto et al. [2006] looked into how bidirectional continuous force input could benefit desktop interaction, such as zooming and scrolling. Hereto, they added an FSR to a trackpad to sense both touch and force of the user's index finger. A piezo-ceramic actuator provided tactile feedback to the user's force input. As

Rekimoto et al. [2006] added bidirectional zooming and scrolling via force to a trackpad.

application examples, Rekimoto et al. showcased zooming a map or scrolling a menu list. The more force the user applies, the further the user scrolls or the more she zooms. The finger pose is used to determine the direction: When the finger lies flat on the trackpad while applying force, its contact size increases, which is interpreted as positive force, e.g., to zoom in. Pressing with the finger normally to the trackpad surface results in a small contact area and is interpreted as negative force, e.g., to zoom out.

Styli on Tablets

Graphic tablets are touch-sensitive tablets that are typically used in combination with a force-sensitive stylus.

Graphic tablets with styli are often used for drawing and sketching. In addition, they are also used as generic absolute pointing device. The tablet, which usually does not have a built-in screen, serves as a giant trackpad that can sense the stylus' touch location in 2D, e.g., to control a mouse cursor. While the tablet itself is usually not force-sensitive, the stylus has a force-sensitive tip that delivers force readings with a typical resolution of about 10 bit. Meanwhile, styli have also become popular for writing and drawing on multi-touch tablets, such as Apple iPad. HCI researchers have looked into the combination of pen strokes and force input as interaction technique for the desktop environment.

G. Ramos et al. [2004] investigated different confirmation techniques for PBLT menu control with a force-sensitive stylus.

G. Ramos et al. [2004] studied PBLT with a stylus on a tablet connected to a screen and found that users can reliably control up to six levels when continuous visual feedback is given. Hiding the visual cursor updates significantly increases error and selection time. Despite the use of a linear transfer function, users found that the system was very sensitive to low force, which indicates that humans underestimate their force input at the lower end of the force space (cf. Section 2.3). The authors also tested different confirmation techniques for force input via the stylus. Although the subjects preferred Quick Release as confirmation technique most, they performed significantly better in selecting items with the force-sensitive pen using a 1 s Dwell Time technique. Drawing a stroke or pressing a button on the stylus to confirm the input instead lead to high selection errors since both techniques interfered with the acquired force. Ramos et al. also showcased a set of widgets, in which force controls the position, angle, or scale of the widget's cursor or target (cf. Figure 3.4).

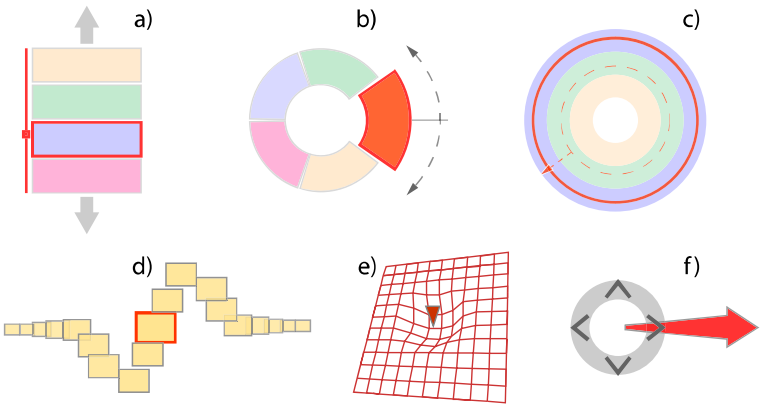


Figure 3.4: Pressure Widgets by G. Ramos et al. [2004]. (a) PBLT-controlled linear menu. (b) PBLT-controlled circular menu. (c) PBLT-controlled concentric circular menu. (d) Horizontal fisheye menu with force controlling the scale of each target. (e) Wire frame widget that becomes pointier the more force is applied to it. (f) Marking menu that allows to map multiple items per stroke depending on how much force is applied. Image taken from G. Ramos et al. [2004].

Forlines et al. [2005] exploited the transient nature of force input to let the user peek into a preview state before committing an action.

Forlines et al. [2005] suggested utilizing binary force input to show the user a transient preview mode to actions before committing them. By pressing softly with a stylus on a tablet or with a finger on a force-sensitive touchscreen, the user peeks the effect of an action, such as changing the text color in a document. If not satisfied, the user simply releases force and the previewed effect is discarded. To confirm instead, the user presses stronger, which commits the action and puts it onto the undo stack. Two examples were showcased: (1) To reposition the camera view in a map, the user drags the pen with slight force and then presses harder to fix the position. Alternatively, the view can be reset when force is released. (2) To preview the magnification of a map, the user applies light pressure to the pen tip. To confirm, a strong press is applied. To restore the magnification level instead, the user simply lifts the pen off the tablet to reduce force.

Y. Li et al. [2005] used force input from a stylus as modifier for fast switching between drawing and gesturing.

Y. Li et al. [2005] investigated mode switching techniques for quick alternation between drawing and gesturing modes for styli on tablets. One of the techniques utilizes force readings from the stylus and divides the force space into three levels. Following G. Ramos et al. [2004], the first level is ignored since humans find

it difficult to control force at the lower end of the force space. When the applied force matches the middle level, the stylus enters the drawing mode, and when it matches the upper level, the UI switches to gesturing mode. While a study showed that this interaction technique enabled fast switching between drawing and gesturing, it led to a higher number of mode errors compared to pressing a button with the non-dominant hand. As a conclusion, the authors suggested adjusting the threshold between the two mode-switching levels based on user-individual calibration.

G. Ramos et al. [2005] utilized force input to let users control the granularity of a slider widget. The more force is applied with the stylus, the finer the slider granularity becomes.

G. Ramos et al. [2005] presented a stylus-controlled slider with adjustable granularity that is controlled via force input: When the user drags the stylus across the tablet surface while applying slight force at the pen tip, the slider cursor moves at coarse granularity. With increasing force the granularity is set to a finer level such that the same dragging distance moves the cursor in smaller steps. Based on this interaction, the authors tested a unimanual pan-and-zoom technique with users. The subjects were not only faster but also preferred this technique over having slider and granularity control split across two hands.

Pressure Marks combine pen strokes with commands that are determined via the force applied to the stylus. This enables quick simultaneous selection and application of commands to multiple objects.

To speed up the task of selecting virtual objects and immediately applying a command to them, G. A. Ramos et al. [2007] merged both steps into a single pen stroke, called a *Pressure Mark*: While the stroke enclosure determines which objects to select, the applied force pattern (constantly low, low-to-high, high-to-low, and constantly high) determines one out of four commands. For example, for the low-to-high pattern, the user draws the enclosure starting at low force and increases the force to the high level before the stroke is finished. In their study, the authors found that users were significantly faster with Pressure Marks compared to the classic sequential approach of selecting objects and applying commands to them. An apt example for Pressure Marks is dragging a set of icons from the desktop screen by enclosing them with a circle stroke drawn with a low-to-high force pattern (Fig. 3.5).

Ren et al. [2007] used force from a stylus to make the selection of small targets that are surrounded by many other small targets

Ren et al. [2007] used force for efficient target selection with a stylus on a tablet for both, large targets in a low density environment, and small targets in a high density environment. In a low density environment, the cursor area grows with force. When it reaches a target, the user lifts the pen to select the target. For a high density environment, force is applied to enable a zoom-

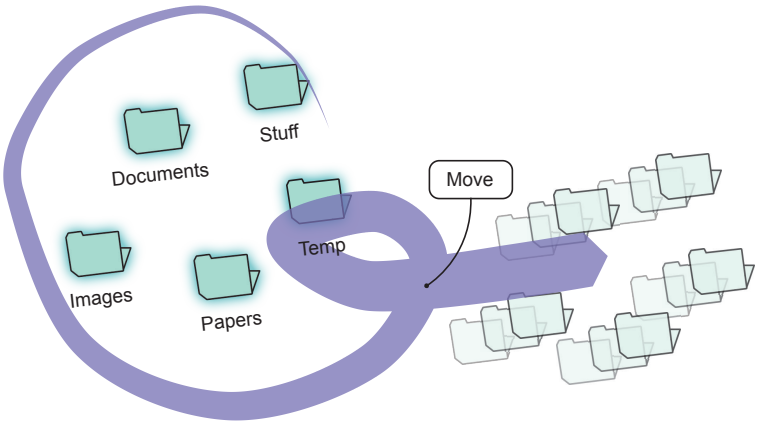


Figure 3.5: Example of a Pressure Mark. The user wants to select multiple folders at the same time and drag them to a new location on the desktop. Hereto, the user draws a stroke around the desired folders that starts with low force and ends with high force. This force pattern encodes the command of selecting all enclosed folders and moving them to the stroke destination. Image based on G. A. Ramos et al. [2007].

ing mode that enlarges targets and spreads them apart to obtain a low density environment. This way, force enables selection of targets without significant movement of the stylus. In their study, the authors found that their technique was especially effective for small targets in high density environments compared to direct selection of targets via the stylus.

more efficient. The more force is applied, the further the targets are spread away from each other.

Büring et al. [2008] designed an interaction technique for simultaneous panning and zooming in maps with a stylus. The stylus' force space is divided into three levels: The lower level is mapped to zooming out of the map at a constant speed, whereas the upper level zooms in. When the stylus force matches the middle level, the current zooming level is maintained. To pan the map, the user simply drags the stylus across the tablet's built-in screen. Although this technique was designed to speed up navigation in maps, a controlled experiment showed that users still performed zooming and panning sequentially.

Büring et al. [2008] used force input from a stylus for constant rate-based zooming of a digital map.

Isometric Joysticks

An isometric joysticks is a small knob that covers four strain gauges. Such input device is often used to steer a cursor in 2D via force.

Isometric joysticks provide relative coordinates, usually in 2D, for controlling a cursor in pointing tasks. They are usually made of strain gauges that sense lateral force applied by the user's finger. The magnitude of force determines the rate, i.e., the velocity, at which the cursor moves, whereas the direction of force controls the cursor direction. Each direction requires a separate strain gauge, hence, for a two-dimensional control, an isometric joystick consists of four strain gauges to steer the cursor either left, right, up, or down. Like all isometric devices, isometric joysticks do not provide kinesthetic feedback because the device cannot be perceptibly moved: the users' fingers and limbs do not move while exerting force on the device [Buxton, 2016].

S. K. Card et al. [1978] investigated text selection via an isometric joystick and compared users' performance against that from mouse input.

S. K. Card et al. [1978] evaluated speed and error rate for text selection via a mouse, keyboard keys, and an isometric joystick. Their isometric joystick consisted of a small rubber knob mounted to strain gauges that allowed cursor control in all four directions across the text. The device was directly placed next to the keyboard, whose keys were used for confirming input. The joystick mapped 4–45 N of force to steer the cursor at linearly increasing velocity. While both, the joystick and the mouse followed Fitts' Law for the positioning time of the cursor, the mouse was clearly superior to all tested techniques as users were both faster and more accurate in selecting text with the mouse.

Rutledge et al. [1990] added an isometric joystick as pointing device between the keys of a computer keyboard to reduce the time otherwise needed to switch between mouse and keyboard.

To minimize the time needed to switch back and forth between mouse and keyboard, Rutledge et al. [1990] designed an isometric joystick that fits on a keyboard. The isometric joystick consists of a tiny steel rod mounted on an acrylic plate and sticks out between the central 'G' and 'H' keys on the keyboard. This *Pointing Stick* enables the user to control a mouse cursor via force in all four directions. The more force is applied, the faster the cursor moves. The authors specifically cared about the design of the transfer function that maps force to the velocity with which the cursor moves across the screen. They found that a linear transfer function with a plateau at the beginning of the force space and a plateau with a speed bump at the very end of the force space worked best for users: the plateaus made controlling the high sensitivity of the strain gauges easier, provided more comfort, and reduced fatigue. For sole pointing tasks users needed 55%

more time with the Pointing Stick than with the mouse. However, users were faster with the Pointing Stick for intermixed pointing and homing to the keyboard. The study data also revealed that further training could increase the speed benefit for the Pointing Stick.

In a more detailed follow-up study, Selker et al. [1991] investigated users' accuracy and precision of controlling the Pointing Stick with an added 3×5 mm finger rest at the top. The authors found that users could control 16–64 different positions per axis when visual feedback was given. Controlling both, the x- and y-axis simultaneously yielded the same performance as controlling each axis individually. Furthermore, there was no difference in user performance regarding whether the Pointing Stick was controlled with a single finger vs. when using a precision grip with the index finger and the thumb. In general, users tended to be more accurate and precise in controlling the direction of force rather than the magnitude of force.

When using an isometric joystick as pointing device, users are more accurate in controlling the direction of force compared to the magnitude of force.

Mithal et al. [1996] found a possible explanation for this: They compared users' movement microstructure for the isometric joystick to the mouse and measured a *tremor* in users' fingers while they were controlling the joystick. Tremor is a series of random variation of force in the user's finger and the rest of the human body that tends to increase the more force the user applies [Stein et al., 2011]. This tremor causes unwanted changes in cursor velocity, which makes it difficult for the user to position a cursor with fine-grained control using an isometric input device. In particular, users had difficulties in placing the cursor in small targets. While it is also likely that users develop a tremor when steering a mouse, the authors concluded that the mouse can dampen tremor very well because of the sliding friction between the mouse and its underlying surface. Furthermore, the user's arm resting on the table also helps absorbing variation in force. For the isometric joystick, damping is very little, if at all. The authors concluded that making the joystick slightly deformable or filtering the tremor out of the captured signal could improve fine-grained cursor control.

Mithal et al. [1996] observed that the user's finger shows signs of a tremor while applying force on an isometric joystick. This leads to a decrease in accuracy of control compared to when using a mouse for pointing tasks.

While Rutledge et al. [1990] initially added an isometric joystick to a keyboard to minimize the homing time between the pointing and the typing device, Zhai et al. [1997] added an isometric joystick, called *TrackPoint*, between the two buttons of a mouse

By adding an isometric joystick between the buttons of a computer mouse, the user can

control two individual mouse cursors to speed up selection of commands: one cursor is used for generic pointing while the other cursor resides at a menu bar, giving the user direct access to the menu without having to move the cursor explicitly beforehand.

to make scrolling, zooming, and command selection more efficient [Ehrlich, 1997]: Whereas the mouse acts as the actual pointing device, the joystick can give quick access to scrolling functionality, e.g., for documents and windows. The direction of force determines the scrolling direction, whereas the magnitude of force determines the scrolling speed. For efficient zooming, the mouse cursor location determines the position of a zooming window, while the isometric joystick enables the user to pan within that window. To speed up the switching time between pointing and picking a command from a menu, the joystick controls a second cursor that is located at menus and toolbars. Rutledge et al. [1990] also demonstrated how two TrackPoints on a keyboard (Fig. 1.5) benefit drawing and text selection [Ehrlich, 1997]: To draw, the user controls the TrackPoint located at the dominant hand, whereas the other hand's TrackPoint is used to pick a drawing tool. For text selection, the user controls the caret with the TrackPoint located near the dominant hand and uses the second TrackPoint to select across different style options with the other hand.

→ Excursus: In-Car Menu Control via Force Input

Another stationary environment, yet different from desktop spaces, is the car cockpit. Similar to the desktop environment, the user is sitting in a chair, with eyes predominantly focused at the frontal windshield. Over the years, car cockpits have been equipped with an increasing amount of electronic devices, such as navigation systems, air conditioning, or digital music players [Horrell, 2014]. Car manufacturers try to unify and centralize their control in multi-purpose rotary knobs, touch pads, and touchscreens [Niedermaier et al., 2009]. Touch interaction, however, can be problematic while driving because the shaking motion of a car on bumpy road makes it difficult to accurately move the finger, e.g., across a virtual slider for temperature control. Compared to touch input, force input has the benefit that the finger does not need to be moved for one-dimensional input and that the increased friction between finger and touch surface keeps the finger better in place. This is why research has looked into force input as a promising candidate for in-car control.

Ng et al. [2016] compared in-car menu control with a physical dial against direct touch input and force-sensitive buttons while users were driving in a car. The buttons allowed users to navigate stepwise through the menu items with every press that exceeds an activation threshold of 3 N. One button moves the cursor up, the other button moves it down. The user confirms her input by pressing both buttons simultaneously. Overall, users were fastest in navigating the menu with direct touch. Using the force-sensitive buttons made them glance more frequently at the display that showed the ten menu items and the cursor. The authors conclude that this interferes with relevant safety issues while driving and that a rate-based mapping for force input might tackle this.

Hence, in a follow-up work, Ng et al. [2017] tested positional vs. rate-based PBLT for a menu with eleven items. While users had problems with positional control, their performance for rate-based control with vibrotactile feedback was as good as using a physical dial.

Sheik-Nainar et al. [2016] compared absolute direct touch input with absolute indirect touch input and relative indirect touch input for two-dimensional menu control in a car simulator. For both indirect methods, a force-sensitive trackpad was used: pressing on the trackpad selects the preselected item. The authors found that absolute indirect touch had comparable efficiency, effectiveness, distraction, and user preference as absolute touch input. However, for this technique, force input is only of minor matter.

Huber et al. [2017] added two force-sensitive trackpads to the left and right of a steering wheel for drivers to access common in-car commands. The authors conducted an elicitation study to reveal the drivers' envisioned gestures for commands like up-/down navigation of a list menu, in-/decreasing climate fan speed, or play/stop music. In general, they found that users consistently used force as initiation for continuous gestures, such as panning and zooming a map and for confirming their input.

Next, we will present interaction techniques for handheld devices that utilize force input.

3.2.2 Handheld Force Input Techniques

Subsequently, we present force-based interaction techniques for handheld device use.

In this section, we highlight how force input augments interaction with handheld devices, such as Personal Digital Assistants (PDAs) with a stylus, smartphones, and tablets, using the dominant hand (DH) and the non-dominant hand (NDH). We group and summarize relevant related work regarding the main benefits that these force-based interaction techniques for handheld devices provide.

Note that we intentionally do not include force-based interaction techniques for wearable devices, like smartwatches, since they are out of the scope of this thesis. For the same reason, force-related manipulations, such as squeeze and deformation, are not discussed.

Secondary "Click"

Unlike touch input, force input enables the user to express multiple, different commands for the same touch location.

The secondary click is a common action used in desktop interaction with the mouse. A computer mouse typically has two buttons. The left button executes a primary command at the location of the mouse cursor, while the right button can be used to execute a secondary command at the same location. A typical example is accessing a context menu. When the user clicks with the left mouse button on the calendar icon in the *macOS Dock* (Fig. 3.6), e.g., the calendar app is launched. Clicking with the right mouse button on that icon, however, opens up a context menu that enables the user, e.g., to create an new calendar event. On handheld devices, which are typically equipped with a touchscreen for input, this efficient trick is not available. Yet, force input can be used to tackle this problem: A light touch can be used to execute the primary command, whereas a force touch is used to trigger the secondary command.

Roudaut et al. [2008] used force input to trigger a mode for easing the selection of

Roudaut et al. [2008] utilized this technique on a PDA to let the user activate an indirect touch selection mode, called MagStick, without sacrificing direct touch input. Touching softly directly selects an underlying target, while a force touch lets the user

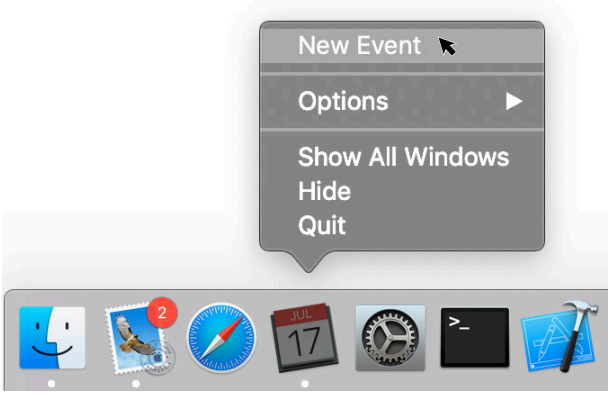


Figure 3.6: Example for a secondary click action used in desktop window systems. The bottom shows the *macOS Dock*, a bar that gives quick access to applications. Clicking with the left mouse button, e.g., on the calendar icon opens the calendar app. However, when clicking on the same icon with the right mouse button, a context menu is displayed. This context menu enables the user to access shortcuts, such as creating a new calender event.

drag a virtual stick from that position in the opposite direction to another target. Hence, this technique enables the user to preselect a target without occluding it with the finger. Upon finger lift-off, the target closest to the stick is automatically selected and the MagStick mode is exited.

distant targets on a handheld device.

Brewster et al. [2009] used touch vs. force touch to discriminate typing with small vs. capital letters on a smartphone with a resistive touchscreen. The authors showed that this technique can lead to faster text input compared to using a second finger that holds a virtual shift key to type capital letters.

Force input can also be used to discriminate typing small vs. capital letters on a handheld device.

Rendl et al. [2016] suggested the same strategy for their FlexCase prototype, a force-sensitive display cover that acts as secondary input space for smartphones.

Arif et al. [2013] utilized the secondary force “click” to enable bypassing inaccurate word predictions within the flow of typing on a smartphone. While typical smartphone keyboards provide word predictions based on the previously typed characters, the user has to actively withdraw them when they do not match because otherwise, when hitting the space bar, the predicted word

Arif et al. [2013] used force to bypass inaccurate word predictions within the flow of typing on a smartphone.

automatically replaces the user's entered characters. To not interrupt the user and make her typing more efficient, the authors' technique allows the user to bypass an inadequate word prediction by force-touching on the next character key that she enters.

Heo et al. [2012] divided the force space of an FSR attached to a smartphone into four different levels, each representing a different mode trigger.

Instead of merely distinguishing a touch from a force touch, Heo et al. [2012] investigated multiple different "click" types. They divided the force space of an FSR attached to a smartphone into four different levels. The first level is mapped to light touch input, whereas the other levels are mapped to stronger force input. This technique saves both display space and time for reaching back and forth between alternative on-screen buttons. Such multi-force button can be used, e.g., to make 3D object interaction on handheld devices more efficient: touch input moves the object in 2D, whereas the force modes move the object along the z-axis, rotate it, or pan the scene.

Goguey et al. [2018] mapped force to different levels of granularity for text selection. For example, a light press selects a single word, whereas a strong press selects an entire sentence.

Goguey et al. [2018] used a force-based mode gauge for controlling text selection at five different levels of granularity with a single finger on a smartphone. Like Heo et al. [2012], the force space was divided into different zones. To select some text, the user first places the carat with her finger and then drags it at low force. Force sets the granularity, ranging from character level over word, sentence, or paragraph level to whole text selection, via the mode gauge. To enlarge the selection, the user maintains the force level and drags her finger. Lifting the finger exits the selection mode. Users significantly preferred this technique over the standard multi-tap and force text selection techniques found in Apple iPhone.

Heo et al. [2011a] used tangential force to extend already used touch gestures with further commands.

Heo et al. [2011a] utilized shear force in x- and y-direction to enable the user to use similar gestures for similar navigation tasks, like navigating a website. For example, to scroll the website to the left and right, the user drags with her finger in the opposite direction. To navigate to the previous or next site in the browser's history stack, the user also sets out to drag to the left or right but this time with added force. Hence, while using a similar gesture for both navigation tasks, the system discriminates the user's intention through the intensity of force.

C. Harrison et al. [2012] extended free-finger drawing

C. Harrison et al. [2012] exploited shear in a similar fashion for drawing and interaction with documents. To draw free-hand strokes on a handheld device, the user drags with light

force across the touchscreen, whereas force-dragging changes to drawing straight lines. To move a document, the user drags it, but when adding force to the dragging motion, the file is copied to the dragging destination.

with a mode to draw straight lines when the user applies shear force while drawing.

3.2.3 Text Input

As already seen by Brewster et al. [2009] and Rendl et al. [2016], force input also benefits text input on handheld devices. It not only benefits efficient input of small and capital letters or bypassing wrong predictions, but also enables entering words faster, or typing on a touchscreen while looking at a distant screen.

McCallum et al. [2009] suggested a force-based alternative to multi-tap text input via the numpad on mobile phones. Classic multi-tap maps three to four characters to a single button on the numpad, with each character being discriminated by a different number of sequential taps. To overcome this tedious and repeated tapping, the authors added an FSR to the device and divided the force space by the number of characters mapped to a key. Instead of multi-tapping on a key, a character is entered depending on how much force the user exerts on the physical button. Using this technique, trained users were about 5% faster compared to typing with multi-tap.

McCallum et al. [2009] presented a force-based alternative to multi-tap text input on a handheld device that are equipped with a numeric keypad for typing.

Zhong et al. [2018] turned the touchscreen of a smartphone into a single, large force-sensitive button that enables the user to type on a distant screen without looking at her smartphone. The distant screen displays the alphabet in a horizontal row, from which the user can select a subset of adjacent characters based on how strongly she presses against the handheld screen. The harder she presses, the further the selection moves towards the end of the alphabet. To enter a word, the user preselects a subset that contains the word's first letter and releases her force to confirm. Using multi-tap, the user can then iterate through the letters of the subset. This procedure is continued until all characters have been entered. To speed this up, the technique can also switch to direct word-level input and let the user multi-tap though a list of predicted words. With this technique, users reached a typing speed of 11 WPM after ten minutes of training.

Zhong et al. [2018] used a force-sensitive smartphone to enable users to type at a distant screen without looking at the input device.

Authentication

Force input can also be used as an "invisible" input layer that adds additional security to entering passcodes on a touchscreen.

Arif et al. [2014] uses force input as additional security layer for entering passcodes on a smartphone. To unlock the device, the code must be entered with a specific force pattern.

Arif et al. [2014] presented this technique to let the user define a passcode for her device while simultaneously capturing the magnitude of force that she applied to each key. This way, not only the correct sequence of digits must be entered to unlock the device, but also the appropriate force pattern must match. In a study, the authors showed that the technique was more resistant to unauthorized persons guessing the correct PIN code by looking at display smudges. However, users required more tries to unlock their device compared to when only using a passcode.

Rendl et al. [2016] adapted the same idea as application example for their force-sensitive FlexCase smartphone cover.

3D Object Manipulation

While the location of touch input on a touchscreen is two dimensional, force can add a virtual third dimension to it that describes how far the touch point is "pushed" into the two dimensional touchscreen plane.

Qiu et al. [2016] presented a force-based interaction concept for handheld devices that enables users to move virtual 3D objects along the z-dimension.

Qiu et al. [2016] utilized this third dimension of touch for object manipulation along the x-, y-, and z-axis on force-sensitive handheld touchscreens. Hereto, they divided the force space into three sections: light touch, normal touch, and strong touch. To move the virtual object along the x- and y-axis, the user drags it with light touch intensity across the touchscreen. When the user taps the object with light touch intensity, it is attracted towards the user, whereas a hard touch pushes the object away along the z-axis. Despite the natural mapping, users were significantly slower with this technique and preferred it less than splitting the control across two hands, where one hand would control the translation of the object along the x- and y-axis, and the other hand the z-axis.

Grip Maintenance

Unlike touch input, entering a value via force input requires no finger displacement.

Using touch to input a one-dimensional value usually requires finger displacement, like dragging a slider knob, or a multi-tap strategy that alters the value incrementally via repeated finger taps. The latter, however, is tedious and time-consuming, and finger displacement is only feasible using the thumb on a handheld device, because all other fingers are statically grasping the device, holding it in place. Force input, however, requires no finger displacement and therefore enables these grasping fingers to be additionally used for input.

B. L. Harrison et al. [1998] added an FSR to the chassis of a PDA that is held with two hands in landscape orientation. The device uses tilt for input, but to prevent accidental input, the user must first press against the chassis to active the tilt sensor.

Hoggan et al. [2012] equipped the side of a mobile phone with an FSR and enabled the caller to convey information to the callee by squeezing the device without losing the grip. Depending on how much force the caller applies, up to four different pressure messages, called *Pressages*, are supported that the callee receives as different vibration patterns during the call.

Wilson et al. [2013] used PBLT to let fingers that are usually just used for holding the handheld device in place also control menus and gestures.

Wilson et al. [2013] added an FSR for each finger involved in grasping a smartphone held in portrait orientation. Using PBLT, these force buttons could be used to control linear on-screen menus for each finger while maintaining a firm device grip. The authors also showcased how the technique makes navigating a website more convenient: Pressing softly, e.g., with the middle finger, scrolls a website upwards, but when pressing harder with that finger, the browser immediately scrolls to the top.

McLachlan et al. [2014] used force input on a tablet device for velocity-based scrolling with the NDH that otherwise would only be used for holding the device.

McLachlan et al. [2013] investigated bimanual scrolling techniques on a tablet device that is held in landscape orientation. While the DH is used to determine the scrolling direction, the hand holding the tablet controls the speed by pressing an FSR mounted to the tablet bezel. The authors found that their technique enables the user to scroll and select targets significantly faster compared to unimanual techniques, where the user controls the scrolling speed through flicking gestures.

In a follow-up work, McLachlan et al. [2014] used force input from the NDH to also control linear menus via PBLT together with touch input.

Similarly, McLachlan et al. [2014] utilized force input via the tablet bezel to let the user access transient modes—or quasi-modes [Raskin, 2000]—that are only active as long as a certain amount of force is maintained. As examples, the authors illustrate a weather forecast application that lets the user peek into each of the upcoming days' forecast via force. Releasing the force automatically flips back to the current day. Another example is a map application that allows the user to temporarily toggle between map-, street-, and satellite view depending on how much force the NDH's thumb applies. In these examples, the DH can then interact with the transient UI via tapping or continuous gestures [McLachlan et al., 2015].

Scrolling and Zooming

Scrolling and zooming via touch is tedious since the gestures must be performed repeatedly. Rate-based force input can tackle this problem.

Interacting with content, such as websites, documents, images, or maps, often requires scrolling and zooming since the touchscreen real estate, especially smartphones, is relatively small. To scroll or zoom, the user typically performs repeated gestures, such as flicking with a finger or pinching with two fingers. This can be tedious and time-consuming when navigating large content. Using force input with rate-based control can significantly speed up the interaction without having to perform repeated gestures.

Quinn et al. [2009] used force-based for efficient zooming into alphabetically ordered lists on a handheld device.

Quinn et al. [2009] used rate-based force input for zooming into alphabetically ordered lists via a stylus on a PDA. By pressing with the pen on a letter from the alphabetic overview, the user zooms into the list starting with that letter. The more force is applied, the faster the zooming speed. For fine-grained navigation within the zoomed state, the user performs flicking gestures with the pen to scroll up or down. Using this technique, users were 15% faster in browsing a list of 1,500 items compared to scrolling by dragging and flicking.

Miyaki et al. [2009] presented a bidirectional rate-based technique for zooming maps on a smartphone.

Miyaki et al. [2009] presented a bidirectional technique for zooming with a single finger on a force-sensitive handheld touchscreen. To zoom in, the user touches the screen at the region of interest and presses. As long as force is maintained, zooming continues. To zoom out instead, the user performs a tiny sliding gesture prior to applying force.

C. Harrison et al. [2012] utilized shear force for rate-based 2D scrolling of content displayed on a handheld touchscreen.

C. Harrison et al. [2012] experimented with tangential force input to enable scrolling and zooming with a single finger via shearing. To scroll two-dimensionally, the user presses the finger and pushed it to the desired direction. The magnitude of the applied force controls the scrolling speed. Zooming works similarly. The direction of force determines whether the user wants to zoom in or out, and, again, the magnitude of force controls the zooming speed.

Spelmezan et al. [2013a] combined an FSR with a proximity sensor to enable rate-based scrolling by pressing with the thumb against the side of a smartphone.

Spelmezan et al. [2013a] added a force- and proximity-sensing button to the side of a smartphone for bidirectional rate-based scrolling. Pressing the thumb on the button scrolls down, whereas putting the thumb away from the button scrolls up. In a follow-up work, Spelmezan et al. [2013b] replaced the proximity sensor with a second FSR. For small scrolling distances, users were still faster using touch gestures, but for large distances they became significantly faster with rate-based force input.

Holman et al. [2013] used a similar scrolling technique. In addition, they also enabled the user to actively slow down scrolling through force via squeezing the side of the phone.

Wilson et al. [2013] used side-mounted FSRs for scrolling websites: a soft press incrementally scrolls the website upwards, but a strong press immediately scrolls to the beginning of the site.

To enable all grasping finger to interact with the smartphone while it is being held with one hand, Wilson et al. [2013] added an FSR for each finger to the side of the smartphone. They illustrated how this setup can be used to scroll websites: Pressing softly, e.g., with the middle finger, scrolls a website upwards, but when pressing harder with that finger, the browser immediately scrolls to the beginning of the site. Bidirectional rate-based zooming is also illustrated: One finger presses to zoom in, and the other finger presses to zoom out while the magnitude of force controls the zooming speed.

Antoine et al. [2017] used rate-based force input for efficient repositioning of items in long lists.

Antoine et al. [2017] targeted repositioning of list items with simultaneous scrolling for smartphones with force-sensitive touchscreens. Hereto, the user first drags the list item that she wants to replace to the bottom of the screen and then presses to start scrolling through the list. The more force is applied, the faster the list is being scrolled down. When force is released, scrolling slows down and the user can drag the list item to the desired location in the list. Users performed faster and committed fewer errors using this technique compared to touch-based solutions, e.g., with flicking.

Pelurson et al. [2016] presented n efficient bimanual rate-based scrolling technique for large data sets that are visualized along a horizontal axis.

For fast scrolling through content that is laid out horizontally, Pelurson et al. [2016] presented a force-based technique that uses both hands to interact with the touchscreen of smartphone that is held in landscape orientation. First, the user sets the scrolling direction by performing a swipe gesture to the left or right with the thumb from the DH. To start scrolling in continuous mode, the user then presses with the NHD's thumb on an FSR that is mounted to the screen bezel. The more force is applied, the faster the scrolling speed. Alternatively, force taps scroll the content in discrete steps: A light tap results in short jumps, whereas a strong tap leads to longer jumps. When the target area has been reached, the user fine-tunes the navigation via drag-flick gestures. Using the continuous mode, users navigated as fast as with direct touch input via repeated flicking gestures. Furthermore, the force-based technique significantly reduced screen occlusion during interaction.

Gain Factor Control

Due to the fat finger problem [Siek et al., 2005], fine-grained manipulations on touchscreens are difficult to perform.

To give the user the illusion of directly interacting with a virtual object when touching it, the control-gain ratio is usually 1:1. However, this 1:1 ratio makes fine-grained manipulations of the virtual object difficult since humans are limited in controlling their fingers in tiny movements and the fat finger problem [Siek et al., 2005] makes it difficult for the system to interpret the user's touch point accurately.

Changing the control-gain ratio via force input can solve this problem.

To solve this problem for handheld touch input, C. Harrison et al. [2012] built a prototype that changes the gain factor to 1:10 when the user exerts force against the touchscreen. Using shear, the user can displace the virtual object in x- and y-direction with fine-grained control without actually moving the finger.

Besançon et al. [2017] used a force-coupled control-gain ratio for coarse vs. precise displacements of virtual objects displayed on a tablet device.

Besançon et al. [2017] presented an interaction technique that allows to reposition virtual objects on a tablet device by moving the tablet in space (Fig. 3.7). For example, to move the object horizontally, the user moves the tablet horizontally in front of her. To adjust the gain factor that affects the granularity of the mapping between the physical tablet displacement and the virtual object displacement, the authors added force sensors to the back of the device. To avoid clutching, the user presses at the

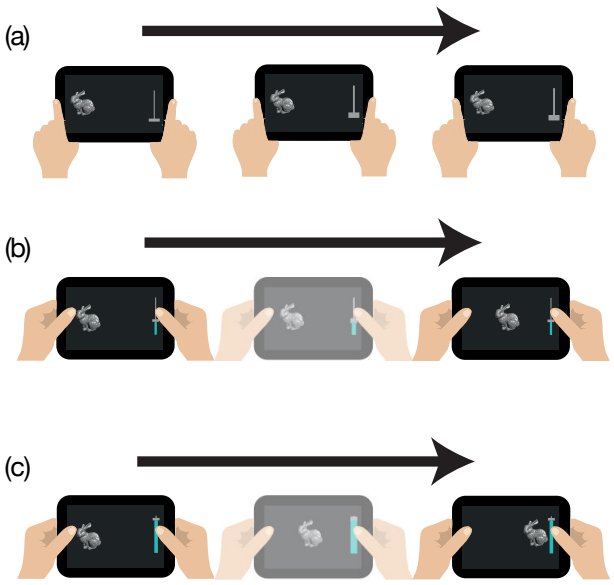


Figure 3.7: Gain factor control via force input at the back of a tablet device. The user wants to reposition a virtual object from left to right. (a) When the user moves the tablet device horizontally in space, the object is only slightly moved to the right. (b) Applying force at the back of the device changes the gain factor: moving the tablet horizontally for the same distance as used in (a) moves the object further to the right. (c) The more force is applied, the more the gain factor is affected: now, the virtual object completely moves to the right when the user moves the tablet horizontally. Image (slightly modified) taken from Besançon et al. [2017].

back of the device. This enables the user to alter the gain factor while displacing the tablet in space. Users associated stronger force with a higher gain. Using this technique, users were more precise compared to using a touch slider to adjust the gain factor.

3.3 Overview: Force Input in HCI

We close this chapter with an overview about force-based interaction techniques from HCI literature. The tabular overview contains 77 papers and aims at making easy comparisons across

Finally, we provide an overview of 77 papers about force-based

interaction techniques from HCI literature.

the researched techniques. We decided on eleven different criteria that characterize force-based techniques and we also present application examples given by the authors of each paper. We use the following legend:

- ***Paper:*** First author name and year of publication.
- ***Control:*** Control mechanism used.
  P: Position-Based Control
  R: Rate-Based Control
- ***Levels:*** How many force levels the user can input.
  cont.: continuous (see '*Res.*' for max. number of values.)
- ***Funct.:*** Transfer function used.
  lin.: linear
  log.: logarithmic
  exp.: exponential
- ***Direction:*** In which direction force navigates the values.
  uni: Unidirectional (only for value increase)
  bi: Bidirectional (value increase and decrease)
  bi (x, y): Bidirectional in x and y dimension (shear force)
- ***Confirm.:*** Confirmation technique used.
  DT: Dwell Time in seconds
  QR: Quick Release
  Thresh.: Thresholding
  Other techniques are mentioned in the table.
- ***Force:*** Type of input force.
  N: Normal force (perpendicular to the sensor surface)
  S: Shear force (tangential force)
- ***Feedb.:*** Feedback modality used.
  A: Auditory feedback
  H: Haptic feedback
  V: Visual feedback
- ***Sensor:*** Force sensor used.
- ***Space:*** Available force space from the sensor (in N).
- ***Res.:*** Sensor resp. controller resolution (in bits).
- ***Device:*** For what device the technique has been designed.
- ***Application:*** For what applications the technique is used.
- A star '*' behind the first author's name marks that the technique uses pseudo-force, i.e., force input is estimated without using a dedicated force sensor.

- A dash '–' indicates that the corresponding feature is not available for the technique.
- A question mark '**?**' symbolizes that it was not evident from the paper how the feature was implemented.

Please find the tabular overview on the subsequent pages.

Having looked at and classified existing interaction techniques that utilize force input, we will now present new techniques that contribute to this space. All techniques exploit the higher expressiveness from force input for a single touch point compared to classic touch input. We will start with a technique that makes two-handed handheld use more efficient by exploiting force input from the fingers that usually rest at the back of the device.

Next, we will present novel handheld interaction techniques that exploit the reduced finger displacement from force input.

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
Antoine 2017	R	cont.	exp.	uni	–	N	V	Apple iPhone 6S	4	?	Smartphone, Trackpad	List scrolling
Arif 2013*	P	2	lin.	uni	Thresh.	N	V	Apple iPhone 4	3	?	Smart-phone	Text input modifier
Arif 2014*	P	2–7	lin.	uni	–	N	H	Google Nexus 4	?	?	Smartphone	Passcode entry
Baglioni 2011	R	2, cont.	lin.	uni	–	N	V	HTC Hero	?	?	Smartphone	Flicking
Besançon 2017	P	cont.	lin.	bi	Release	N	V	FSR	?	?	Phablet	Gain factor control
Blaskó 2004	P	2	lin.	bi	–, Thresh.	N	V	Synaptics TM41P-200	?	?	Touchpad	Modifier
Boring 2012	P, R	2, cont.	lin.	uni, bi	Release	N	V	Apple iPhone 4S	?	?	Smartphone	Map panning, map zooming
Brewster 2009	P	3	lin.	bi	DT 0.5 s, QR	N	V	Nokia N800	?	8	Phone, PDA	Text input modifier
Büring 2008	R	3	lin.	bi	QR	N	V	Toshiba Portégé Tablet PC with stylus	?	8	Tablet, Stylus	Map zooming
Buxton 1985*	P	2, 4–8	lin.	bi	–, Thresh.	N	V	none particular	?	?	Capacitive Touch Tablet	Painting, Modifier
Card 1978	R	cont.	lin.	bi (x, y)	–, Key press	S	V	Strain gauge	45	?	Isometric joystick	Text selection
Cechanowicz 2007	P	4, 6, 8, 10, 12, 16, 64	lin.	bi	DT 1 s, QR, Mouse click	N	V	CUI #IESR-R-5L (FSR)	1.5	10	Mouse	Menu control

Figure 3.8: A comparison of force-based interaction techniques from HCI literature (1/9).

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
Corsten 2017	P	3, 5, 7	lin.	bi	Tap	N	V	Apple iPhone 6S	4	9	Smartphone	Menu control, emoji input, gaming
Corsten 2017	P	5, 10, 15	lin.	bi	DT 1 s, QR	N	V	Apple iPhone 6S	4	9	Smartphone	Menu control
Corsten 2018	R	cont.	lin.	bi	–	N	V	Apple iPhone 6S Plus	4	9	Smartphone	Value input
Corsten 2019	P	1–7	log.	bi	QR	N	V	Apple iPhone 6S Plus	4	9	Smartphone	Reaching distant targets
Forlines 2005	P	2	lin.	bi	–	N	V	Tablet PC, Stylus, DiamondTouch	?	8	Tablet with stylus, Tabletop	Color selection, volume control, scrolling
Goel 2011	P, R	3	lin.	uni	–, DT, Thresh.	N	V	Samsung Nexus S, FSR	?	?	Smartphone	Map zooming, text input, modifier, squeezing
Goguey 2018	P	5	lin.	bi	QR	N	V	Apple iPhone 6S	4	?	Smartphone	Modifier, selection granularity
Harrison 1998	P	2	lin.	uni	Thresh.	N	V	Pressure strips	?	?	PDA	Modifier

Figure 3.9: A comparison of force-based interaction techniques from HCI literature (2/9).

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
Harrison 2012	P, R	cont.	lin.	bi (x, y)	–	S	V	Analog joysticks	2 mm shear	9	Custom-built handheld	Modifier, short cuts, panning, zooming, granularity control
Heo 2011* (MobileHCI)	P	2, cont.	lin.	uni	Thresh.	N, S	–, V	Apple iPhone 3GS (accelerometer)	?	?	Smartphone	Modifier, instrument plating, gaming
Heo 2011 (UIST)	P	2 (x, y)	– (thresh.)	uni	Thresh.	N, S	V	FSR	?	10	Smartphone	Force gestures, object control, browsing, navigation,
Heo 2012	P	3, 4, 5	lin.	bi	DT 1 s, drag gesture, roll gesture	N	V	Interlink FSR-400	20	8	Smartphone	Modifier
Herot 1978	P, R	cont.	lin.	uni, bi	–	N, S	V	BLH (SPB3-35-500) strain gauge	?	8–12	Touchscreen (CRT)	Drawing, object movement and rotation
Hoggan 2012	P	4	lin.	bi	QR	N	H	FSR	6	8	Smartphone	Vibrotactile message transmission

Figure 3.10: A comparison of force-based interaction techniques from HCI literature (3/9).

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
Holman 2013	R	cont.	lin.	bi	Decele-ration	N	V	CUI #IESR-R-5L (FSR)	1.5	10	Smartphone	scrolling, zooming, app switcher, text formatting
Huber 2017	–	2	lin.	–	Button press	N	V	Force-Touchpad	8	?	In-car	Accessing commands in combination with touch gestures
Hwang 2012*	P	2, 3, 4, 5, cont.	lin.	bi	Tap	N	V	Apple iPad, iPad 2, iPod, iPhone 4	?	9	Smartphone, Tablet	Menu control, shutter button, gaming
Kuribara 2015	P	cont.	lin.	bi	–	N	V, passiv e H	Interlink FSR-401	10	?	Mousepad	Window management, C-D ratio control
Lee 1985*	P	?	lin.	bi	–	N	–	Capacitive sensor matrix	?	?	Tablet	–
Lee 2012	P	4, 5, 6, 7, 8	lin.	bi	DT 1 s	S	V	Tekscan FlexiForce Model-A201	4.4	?	Smartphone	Map scrolling, map zooming, 3D object manipulation
Mascaro 2003*	P	cont.	lin.	bi	–	N, S	V	Fingernail sensor, 3-axis force sensor	2.25 (S), 3 (N)	?	Custom (stationary)	(Technical study: data collection)

Figure 3.11: A comparison of force-based interaction techniques from HCI literature (4/9).

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
McCallum 2009	P	3	log.	bi	QR	N	V	CUI #IESR-R-5L (FSR)	1.5	10	Custom (handheld)	Text input
McLachlan 2013	R	cont.	lin.	bi	–	N	V	FSR	4, 6, 9	?	Tablet	Scrolling
McLachlan 2014	P	5, 7, 10	lin.	bi	Tap	N	V	Interlink FSR	10	?	Tablet	Menu control
McLachlan 2015	P	5, 7, 10	lin.	bi	Gesture	N	V	Interlink FSR	10	?	Tablet	Menu control, emoji input
Minksy 1984	P	2, cont.	non-lin.	bi	–, Thresh.	N, S	V	Strain gauge	44	12	Touchscreen (CRT)	Finger painting, modifier
Mithal 1996	R	dis-cont.	non-lin.	bi (x, y)	–	S	V	IBM Track-Point II	?	?	Isometric joystick	Pointing
Miyaki 2009	R	cont.	lin.	bi	Release	N	V	Interlink FSR	?	8	Smartphone	Scrolling, zooming
Mizobuchi 2005	P	10	lin.	bi	DT 1 s	N	–, V	Hokuriku HFD-500 (FSR)	4	?	PDA with stylus	Menu control
Ng 2016	P, R	10, cont.	lin.	bi	DT 1.5 s, Button press	N	V, H	Peratech FSR	8	?	In-car	Menu control
Ng 2017	P	2	lin.	bi	Button press	N	V, H	Peratech FSR	?	?	In-car	Menu control
Omata 2007	P	4, 6, 8	lin.	bi	DT 1 s, Mouse click	N	V	phidgets.com force button	29	10	Mouse	Menu control, semantic zooming,

Figure 3.12: A comparison of force-based interaction techniques from HCI literature (5/9).

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
Pelurson 2016	R	2, cont.	lin.	bi	–	N	V	Interlink FSR-400	?	?	Smartphone	Horizontal scrolling
Qui 2016	R	3	non-lin.	bi	–	S	V	?	?	?	Smartphone, Tablet	3D object positioning
Quinn 2009	R	cont.	non-lin.	uni	–	N	V	HP Compaq tc4400 Tablet PC with stylus	?	?	Tablet with stylus	Zooming
Quinn 2019*	P	3	lin.	bi	DT 5 s	N	–	Google Pixel 2, Pixel 2 XL (barometer) Apple iPhone 7, iPhone X	5	?	Smartphone	(Technical study: data collection)
Raisamo 1999	R	2	non-lin.	bi	–, Release	N	V	Elo 20" IntelliTouch	?	8	Touchscreen	Kiosk system interaction
Ramos 2004	P	4, 6, 8, 10, 12	lin.	bi	DT 1 s, QR, Pen click, Pen stroke	N	V	Wacom Intuos	?	10	Tablet with stylus	Menu control, scaling, positioning, rotation
Ramos 2005	P	cont.	exp.	bi	QR	N	V	Wacom Cintiq 18SX with stylus	?	10	Tablet with stylus	Granularity control for sliding and zooming
Ramos 2007	P	2, cont.	—	bi	Gesture	N	V	Toshiba Portégé M200Tablet PC with stylus	?	?	Tablet with stylus	Object selection, command application

Figure 3.13: A comparison of force-based interaction techniques from HCI literature (6/9).

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
Rekimoto 2006	P, R	2, 3, cont.	lin.	bi	–	N	H	Resistor Sensitive Polymer Film	?	?	Touchpad	Scrolling, zooming
Ren 2007	P	cont.	lin.	bi	QR	N	V	Wacom Cintiq 21UX with stylus	?	10	Tablet with stylus	Selection of dense and far objects
Rendl 2012	P	cont.	lin.	bi	–	N	V	PyzoFlex foil (custom)	30	12	Tablet	?
Rendl 2016	?	3, cont.	lin.	?	?	N	V	Piezo-electric film (custom)	?	?	Smartphone	Text input modifier
Rosenberg 2009	P	cont.	near-lin.	bi	–	N	V, A	IFSR (custom)	49	12	Pad/Tablet	3D object control, modifier
Roudaut 2008	P	2	lin.	uni	Thresh.	N	V	HTC P3600 PDA	?	?	PDA	Modifier
Rutledge 1990	R	cont.	near-lin. (plateau)	bi (x, y)	–	S	V	Semiconductor strain gauge	?	?	Isometric Joystick	Pointing
Sasaki 1984*	P	?	lin.	bi	–	N	A	Conductive strips matrix	?	6	Touch Tablet	Volume control
Selker 1991	P	cont.	near-lin. (plateau)	bi (x, y)	–	S	V	Semiconductor strain gauge	2.2	?	Isometric joystick	Pointing
Sheik-Nainar 2016	–	2	lin.	–	Button press	N	V, H	Synaptics ForcePad	5	?	In-car	Menu confirmation
Shi 2008	P	6, 8, 10, 12, 16	lin., clustered, fisheye	bi	DT 0.75 s, Mouse click	N	V	CUI #IESR-R-5L (FSR)	1.5	10	Mouse	Menu control

Figure 3.14: A comparison of force-based interaction techniques from HCI literature (7/9).

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
Shi 2009	P, R	15 (P), 720 (R)	lin., fisheye	bi	DT 0.75 s, Mouse click	N	V	CUI #IESR-R-5L (FSR)	1.5	10	Mouse	Object rotation
Spelmezan 2013 (UIST)	R	cont.	lin.	bi	–	N	V	Interlink FSR-400 Short	?	10	Smartphone	Scrolling
Spelmezan 2013 (MobileHCI)	P, R	2, cont.	lin.	bi	–	N	V	Interlink FSR-400 Short	?	10	Smartphone	Scrolling, zooming
Steward 2010	P	9, cont.	lin., quadratic	bi	DT 1 s	N	V, H	Interlink FSR	?	8	Smartphone	(Technical study: technique comparison)
Steward 2012	P	cont.	lin.	bi	–	N	–	FSR	3	8	Smartphone	Inadvertent force measurement
Takada 2019*	P	2, 3, 4, 5, 6, cont,	lin.	bi	DT 1 s	N	V	Alps Electric HSPPAD038 Barometer (Sony Xperia Z5 Compact, Xperia Z5), LG G Watch R, Interlink FSR-402	12	?	Smartphone, Smartwatch	Music control, tangible interaction
Watanabe 2012*	P	cont.	non-lin.	bi	–	N	V	640 x 480 Webcamera, iPod Touch	5	8	Smartphone	Use finger as joystick

Figure 3.15: A comparison of force-based interaction techniques from HCI literature (8/9).

Paper	Control	Levels	Funct.	Direction	Confirm	Force	Feedb.	Sensor	Space	Res.	Device	Application
Wilson 2010	P	4, 6, 8, 10	lin.	bi	DT 1 s, QR	N	V, A	Nokia N810	?	10	Smartphone	(Technical study: technique comparison)
Wilson 2011	P, R	4, 6, 8, 10	lin.	bi (loop-through)	–, DT 1 s, Release	N	V, A	Interlink FSR-402	4	?	Tablet	(Technical study: technique comparison)
Wilson 2012	P	6	lin.	bi	DT 1s	N	V	Interlink FSR-400	3.5	?	Smartphone	Menu control
Wilson 2013	R	cont.	lin.	bi	DT 3 s	N	V	Interlink FSR 400	3.5	?	Smartphone	Rotation, zooming
Y. Li 2005	P	2, cont.	lin.	bi	DT 1 s, QR	N	V	Toshiba Portégé Tablet PC with Stylus	?	8	Tablet with stylus	Modifier, Line thickness
Zhai 1997	R	dis-cont.	non-lin.	bi (x, y)	–	S	V	IBM TrackPoint	?	?	Isometric joystick	Pointing
Zhong 2018	P	cont.	lin.	bi	DT 0.3 s, QR	N	V	Apple iPhone 6S	3.7	9	Smartphone	Text input

Figure 3.16: A comparison of force-based interaction techniques from HCI literature (9/9).

4

Adding Expressiveness to Handheld Touch Input with Back-of-Device Finger Force

→ Summary

When people hold their smartphone in landscape orientation, they use their thumbs for input on the frontal touchscreen, while their remaining fingers rest on the back of the device (BoD) to stabilize the grip. We designed *BackXPress*, a new interaction technique that lets users create BoD force input with these remaining fingers to augment their interaction with the touchscreen on the front: Users can apply various force levels with each of these fingers to enter different temporary "quasi-modes". Both thumbs can then interact with the frontal screen in these modes. We illustrate the practicality of BackXPress with several sample applications, and report our results from three user studies: Study 1 found that index, middle, and ring finger from both hands are practical to exert BoD force. Study 2 revealed how force touches from these fingers are distributed across the BoD. Study 3 found that users achieved up to 92% accuracy for three separate force levels.

Publications: The work presented in this chapter has been done in collaboration with Bjoern Daehlmann, Simon Voelker, and Jan Borchers. The author of this thesis developed the research idea and relevant research questions, including the motivation of the work. Furthermore, he has designed all experiments and analyzed their data. Most of this work has been published in the Proceedings of ACM CHI 2017 [Corsten et al., 2017a]. The author of this thesis is the main author of the paper; most sections in this chapter are taken from this publication. Part of this work has been published as Bachelor's Thesis [Daehlmann, 2016].

4.1 Motivation

Besides touch interaction at the front of handheld devices, HCI research has also looked into touch interaction at the back of the device (BoD).

Interaction on handheld devices, such as smartphones and tablets, is usually done by touching the frontal screen that serves both for input and output. In recent years, research has not only looked into input at the front, but also at the back of such mobile devices. This back-of-device interaction (BoDI) complements classic touch interaction at the front in two ways: First, it mitigates the occlusion problem, since the finger touching from behind does not occlude the visual output at the front [Baudisch et al., 2009; Wigdor et al., 2007]. Second, it enables the use of more fingers for interaction, since when holding the device in portrait orientation, usually only thumb and/or index finger interact at the front, and in two-handed landscape orientation, only the thumb(s) can interact at the front. In contrast, more fingers are available to touch at the back, e.g., for text input [Buschek et al., 2014; Schoenleben et al., 2013].

BoD interaction is typically based on the location of touch points, while the intensity of each touch point, i.e., the force, has not been exploited for input so far.

So far, most BoDI research has only used the location information of touches at the back [Baudisch et al., 2009; Shen et al., 2009; Wigdor et al., 2007]. However, touch input has more properties than its location [Wang et al., 2009b]. One of these properties is force. Force input on touchscreens has been studied since the 1980s [Buxton et al., 1985]. Even commercial products, such as Apple's iPhone Xs, embed force input into the UI, e.g., by letting the user peek into content while pressing against the touchscreen. In both examples, the user exerts force *transiently* to enter different "quasi-modes" that revert upon force release—hence, no need for displaying a back button. While transient force has been investigated for input at the front [McLachlan et al., 2014], it has not been explored for BoDI.

We designed *BackXPress*, an interaction technique that lets users apply force with their fingers resting at the BoD to augment the interaction with the frontal touchscreen. For example, finger

We designed a new interaction technique called *BackXPress* that utilizes transient force at the back of landscape-oriented smartphones to complement touch input at the front. Landscape orientation is convenient, e.g., for gaming, interaction with media, such as photos and videos, or two-thumb typing. Compared to BoDI in portrait orientation, landscape orientation provides more stability in holding the device—especially important when exerting force—and has the advantage of having up to eight fingers available for force input at the BoD and two thumbs for touching at the front. An application example for BackXPress

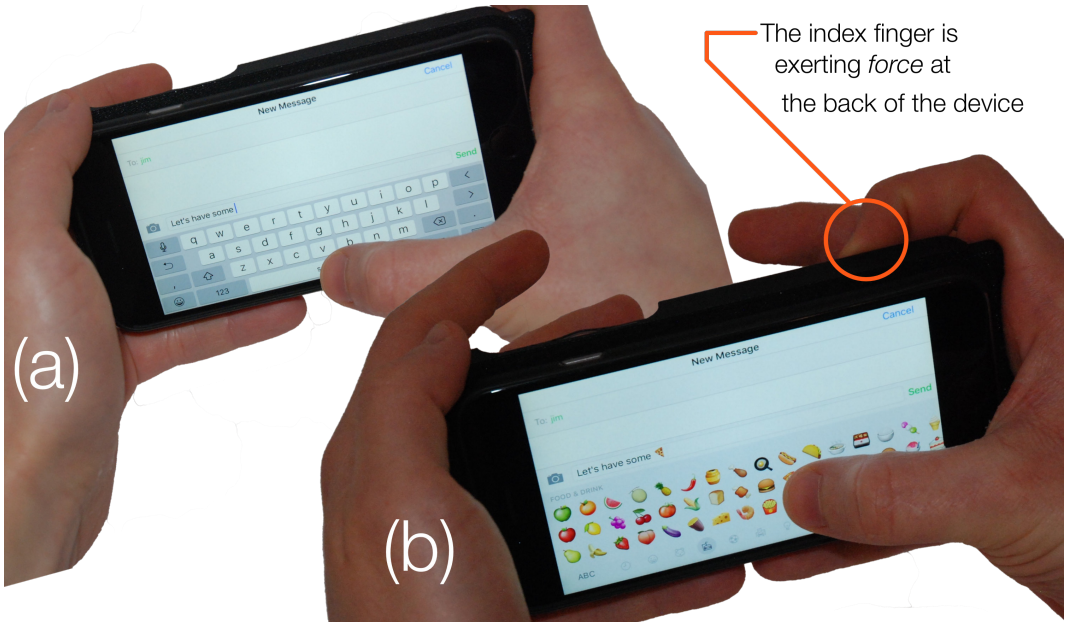


Figure 4.1: Emoji insertion with BackXPress: (a) The user is typing a message on a smartphone keyboard using both thumbs. (b) By exerting back-of-device force, the keyboard switches to emoji layout. Maintaining the force and tapping an emoji appends it to the message. Releasing force reverts to the text keyboard, and the user can continue typing immediately.

is insertion of emoji characters during text entry on a smartphone (Fig. 4.1): When BoD force is applied, the keyboard could switch to show emojis, with each combination of finger and force level showing a different category, e.g., smileys or animals. Tapping an emoji with the thumb would insert it, and subsequent force release could automatically flip back to the regular keyboard to continue typing immediately. With BackXPress, selecting a mode does not occlude the content, since the user's fingers interact at the back. In addition, there is no need for moving the finger to a back button since quasi modes are exited upon force release. Furthermore, the display of controls or menus to access the quasi modes could be omitted for users who are familiar with the finger-to-menu mappings.

force at the BoD can be used to let the user conveniently flip trough different on-screen keyboard layouts.

We investigated this new interaction technique with two studies on device grip and finger placement for BoD force control, and an experiment on users' performance using BackXPress for two use cases: Applying BoD force during single touches at the front, and for up to 20 seconds while touching multiple consecu-

We conducted three user studies to inform the design of BackXPress and evaluate the technique

with users, who achieved more than 92% selection accuracy.

tive frontal targets. Using a correction model, force control accuracy lay above 92% for three separate force levels above normal resting force. We conclude with design guidelines for this new interaction technique.

4.2 Related Work

BackXPress combines **(1)** BoDI on handheld touchscreens such as smartphones and tablets, and **(2)** force input on such devices. We review the related work for both fields below.

4.2.1 Back-of-Device Interaction (BoDI)

First BoD interaction (BoDI) addressed typical occlusion of touch targets by the user's finger.

BoDI improves interaction with handheld devices in various ways. Initial BoDI addressed target occlusion by the finger while touching the screen at the front. For example, LucidTouch [Wigdor et al., 2007] video-captured finger input at the BoD to display the finger silhouette *behind* the target on the frontal screen. NanoTouch [Baudisch et al., 2009] solved the occlusion problem for screens as small as 0.3".

BoDI also compensates target reachability issues on smartphones by splitting touch input across front and BoD.

BoDI also solves reachability issues, i.e., when not all areas at the front can be touched easily, by sensing finger positions at the BoD. For one-handed smartphone usage, Bergstrom-Lehtovirta et al. [2014] predicted the area on the touchscreen that the thumb can reach by—among other parameters—locating the index finger at the back. Löchtefeld et al. [2013] proposed to use the thumb to interact with the lower part of the touchscreen at the front, whereas the index is used to interact at the back as it can reach the upper part of the device easier. Yang et al. [2009] used a similar separation of input but added BoD cursor control to avoid the fat finger problem [Siek et al., 2005]. Hakoda et al. [2015] attached a small photo reflector to the back of a smartphone that, when tapped by the index finger, moved the content shown on the touchscreen towards the area reachable by the thumb. Mohd Noor et al. [2014] improved touch input accuracy by predicting where a touch will land on the screen based on how fingers are grasping the BoD. BoDI also helps to reduce shoulder surfing for passwords entered on smartphones [De Luca et al., 2014].

The examples above address one-handed BoDI in portrait orientation. In contrast, holding the device with two hands provides a more stable grip and has the benefit that up to eight fingers can be used for interacting with the BoD. BrailleTouch [Southern et al., 2012] enabled blind users to type braille characters with six fingers on a flipped smartphone, and Buschek et al. [2014] and Schoenleben et al. [2013] combined a sandwiched tablet with finger pose estimation for typing with eight fingers at the back and two thumbs at the front. BeyondTouch [Zhang et al., 2015] detected up to four different tap locations at the back of a landscape-held smartphone, e.g., to watch and navigate videos. HaptiCase [Corsten et al., 2015] added tactile landmarks to the back of a smartphone that users sense with their fingers for more accurate eyes-free touch input. Shen et al. [2009] used BoD multi-touch on a sandwiched smartphone to control virtual 3D objects.

When holding the handheld device with two hands in landscape orientation, up to eight fingers can be used for BoDI.

BackXPress also exploits the expressiveness of being able to use multiple fingers at the BoD in two-handed landscape orientation. Unlike the examples presented above, our technique allows each finger to express multiple states per finger by force, thus without moving them.

BackXPress captures force input from multiple fingers at the BoD.

4.2.2 Force Input

Force input makes interaction with handheld devices more efficient. Brewster et al. [2009] used binary force on a resistive touchscreen as modifier to type lowercase and uppercase letters without having to reach for a shift key. Forcetap [Heo et al., 2011b] distinguished gentle taps from strong taps with 90% accuracy based on accelerometer data. The simulated force was used to quickly pop up context menus or enable magnification for more accurate acquisition of small touch targets. ForceDrag [Heo et al., 2012] used pressure to toggle a dragging mode on smartphones.

Force input makes handheld interaction more efficient.

Force also adds richer expressiveness to gestures on touchscreens. Davidson et al. [2008] and Qiu et al. [2016] used the force property of a touch to push virtual objects along the z-dimension, and Force Gestures [Heo et al., 2011a] combined normal and tangential force with existing touchscreen input to ob-

Force lets users modify touch gestures to increase their expressiveness.

tain a richer gesture set. Force also enables expressive input even when the fingers are in a static pose, e.g., when holding the device. Spelmezan et al. [2013a] attached a continuous force-sensitive button to the side of a smartphone to control interface widgets by the thumb. Two buttons at the side enabled bidirectional gesturing, such as scrolling and pinching, without finger movement, so that device grip was maintained [Spelmezan et al., 2013b]. Wilson et al. [2013] investigated multi-digit force performance on a smartphone using multiple force sensing resistors (FSR) around the device. Index finger, ring finger, and the combined use of ring and little finger as well as index and middle finger handled force input most accurately, with errors rates as low as 3.2%. For tablets, one hand usually just holds the device. To allow bimanual interaction, McLachlan et al. [2014] added an FSR to the bezel that is covered by the thumb of the hand holding the tablet. Force selected different entries from a menu, while the index finger of the dominant hand touched the screen to interact in that mode. Maintaining force while tapping with the dominant hand reached 96% accuracy, but dropped by 10% when executing sliding gestures.

So far, force input at the BoD has rarely been explored.

To our knowledge, only Stewart et al. [2010] have previously looked into BoD force input. They showed that users could control nine different force levels with the index finger by pushing against the BoD to move a cursor along a horizontal line. BackXPress is the first technique that combines BoD force input with frontal touch on handheld touchscreens to allow multi-digit, multi-level input in a static finger pose.

Transfer Function and Selection

Interaction techniques that use force input are determined by various parameters.

Force-based interaction techniques range in the mapping from sensor values to input values, called *transfer function*, as well as force space and number of targets within that space, selection of force, feedback modality, and context of use.

The design and choice of these parameters significantly influences users' performance for the

Transfer functions vary from linear [G. Ramos et al., 2004], to quadratic [Cechanowicz et al., 2007], to sigmoid [Ren et al., 2007]. Stewart et al. [2010] compared these and stated that a linear mapping worked best. This way, users can control eight to ten levels on a 3–10 N force range in a stationary context

with visual feedback [Cechanowicz et al., 2007; McLachlan et al., 2014; Mizobuchi et al., 2005; Stewart et al., 2010; Wilson et al., 2010]. Walking, however, significantly increases error, selection time, and subjective workload of force input [Wilson et al., 2011]. Target force is typically selected either upon crossing a threshold [Brewster et al., 2009; Heo et al., 2011b], maintaining force for a particular dwell time (usually 1 second), or upon quick release of force [G. Ramos et al., 2004; Wilson et al., 2010; Wilson et al., 2011]. Each of these techniques has certain drawbacks [McLachlan et al., 2014]: Thresholding is limited to two states, dwell time slows down the interaction and does not allow the user to linger on a state longer than the dwell time [Brewster et al., 2009; McLachlan et al., 2014], and quick release causes selection errors [Wilson et al., 2011].

force-based interaction technique.

Transient Force

An alternative is to decouple force selection from its control, e.g., by spitting the two across both hands [McLachlan et al., 2014; McLachlan et al., 2015]. Force can then be modeled as transient [McLachlan et al., 2014]: The user applies force to traverse different force targets. Upon release, targets are revisited in reversed order until the original state is returned. This combination of *natural inverse* and *bounce back* [Ghazali et al., 2005] "encourages the exploration of unfamiliar options and assures the user that errors can be undone" [McLachlan et al., 2014]. When the desired force value is found, the user can then explicitly perform its selection, e.g., with the other hand.

Force is transient, i.e., the user must actively exert and maintain it; when force is released, the system reverts to the state before force was applied.

BackXPress also decouples force selection from its control by letting the user control transient force at the back and selecting the current force target by tapping at the front. Hence, BackXPress also does natural inverse and bounce back, thus enabling users to easily explore and undo force control.

With BackXPress, the user confirms the transient force by tapping the frontal touchscreen.

4.3 The BackXPress Interaction Technique

When the user is holding her smartphone with two hands in landscape orientation, up to eight fingers are in contact with the

BackXPress splits the BoD into rough areas—each for one finger—that allow the user to exert force while maintaining a stable device grip.

BoD, whereas both thumbs can interact with the touchscreen at the front (Fig. 4.2). Unlike existing BoDI, *BackXPress* does not use fine-grained location information of input at the back. Technically, BackXPress divides the BoD into different force-sensitive areas, one for each finger. This way, the user does not need to move the finger and can maintain a stable grip and still communicate rich input: Each finger can apply individual transient force, resulting in a large number of input states, depending on how many different force levels each finger can control. Each finger and force level combination enters a "quasi-mode" as long as force is maintained. The user then interacts in that mode using both thumbs at the front. Upon force release, the mode is exited, and according to *natural inverse*, previously passed modes are quickly traversed in reversed order.

BackXPress targets BoD force maintained for a single tap (1-Tap) or for multiple taps (N-Tap) at the front to confirm input.

BackXPress targets at two different use cases: (1) **1-Tap** and (2) **N-Tap**. In 1-Tap, the user maintains force shortly to tap a single item on the touchscreen in the currently active mode, whereas in N-Tap, the user maintains force over a longer duration to tap multiple consecutive targets at the front. Below, we present three application examples for 1-Tap and N-Tap.

4.3.1 Application Examples

Text Input

An example for 1-Tap is the insertion of an emoji from the emoji keyboard that is activated by BoD force.

(1-Tap): During text entry, the user may want to insert an emoji. Each finger could control a different emoji category, e.g., pressing the left ring finger would display animals. Usually, not all emojis from one category fit on the screen altogether. Different force levels could be used to flip trough different pages. Tapping an emoji by thumb inserts it, and releasing force restores the regular keyboard layout to continue text input.

Entering a whole word with an on-screen keyboard that is different from the default layout is an example for N-Tap.

(N-Tap): The user might need to enter a few word in a language different from the rest of the text. To activate the keyboard that brings the right layout and auto-correction dictionary, the user exerts BoD force to flip through her installed keyboards that are shown on top of the standard layout. When the right keyboard appears, the user maintains force and types the word with both thumbs. When force is released, the standard layout flips back,

and the user can continue typing immediately in her natural language.

Gaming

(1-Tap): In mobile point-and-"click" adventure games, the user frequently engages with an on-screen menu to pick an action, such as *speak* or *walk*, or select an object from the character's inventory. Once selected, the user then taps on the screen to interact with the game context in that mode. With BackXPress, the user could flip through the different actions using force from one finger, whereas another finger presses to select an object. E.g., to speak to a character in the game scene, the user would apply force until *speak* is selected and then tap on the character with the thumb to start the conversation.

Point-and-"click" adventure games are also an apt example for BackXPress' 1-Tap use case.

Hotel Finding

(1-Tap): An application could show nearby hotels on a map of a location at interest. BackXPress could enable the user to explore hotels and experiment with star ratings and price ranges. Force applied on the left side at the BoD controls the hotel ratings between one and five stars. More force pushes the rating down. On the right hand side, the user could then press to control the price range. Pressing harder shrinks the price. Tapping on a "show results" button allows the user to release force and browse all matching hotels in a list.

Using 1-Tap, users can efficiently interact with productivity apps, such as quickly finding nearby hotels with certain star ratings.

As all examples show, with BackXPress the user does not need to move her fingers back and forth between on-screen menus and a back button. The user can maintain a stable grip and still access multiple quasi-modes directly.

4.4 Study 1: BoD Force Finger Candidates

To find out with which fingers users can comfortably apply force at the BoD, we conducted a small user study. We asked 12 participants (19–31 years, M = 22.92, SD = 3.34, two females, all

In Study 1, we wanted to understand which handheld sizes and orientations as well as which fingers are suitable for BoD force input.

right-handed) to press at ten different locations on the back of 4.7" and 5.5" smartphone mockups that were made from acrylic with rounded edges and an enclosing bumper (Fig. 4.2). Users held the mockups in portrait and in landscape orientation. Since the devices were transparent, users could see the targets from the front but also check that their finger position at the back actually landed on a target. Targets were read out by a computer voice, and once a target had been pressed, the instructor initiated the next target. Users were not limited in how to grasp the device and which fingers and hands to use to press the targets. Targets were randomized and repeated once after all targets had been pressed. In total, each user did 2 × 2 × 10 × 2 (Device Size × Orientation × Target × Repetition) = 80 trials. Afterwards, users were asked to state which hands and fingers they considered feasible for BoD force input.

Results

Holding the handheld device with one or two hands in portrait orientation is impractical for exerting BoD force.

In portrait orientation, almost all users used the right hand, whereas only half of the participants considered also the left hand suitable for BoD force input. For the 4.7" mockup, the right index finger was chosen by eleven subjects, but only seven users used the left index finger. Other fingers did not exceed a count of four, with the little finger just a count of one. These results were almost similar for the 5.5" device (Fig. 4.3). During the study, we observed that eight participants preferred to use just one hand to both grasp the portrait mockups and apply force. Using this grip, however, users had to frequently tilt and re-grasp the device to stabilize their grip for reaching lower targets. When two hands were used, fingers snagged. Therefore, we conclude that portrait orientation is rather impractical for BoD force.

Using a 4.7" form factor held in landscape orientation was the most practical to exert BoD force with index, middle, and ring fingers from both hands.

For both landscape form factors, users applied the typical hand posture that people use when interacting with a smartphone in landscape orientation: Both hands hold the device, with the thumbs resting above the front, and the remaining fingers resting at the back to stabilize the device. When users exerted BoD force for a target, they tended to gently push the hands against the device side to obtain additional stabilization. For the 4.7" device, nobody opted for the little finger to apply BoD force; index and middle finger were most preferred (ten resp. eleven

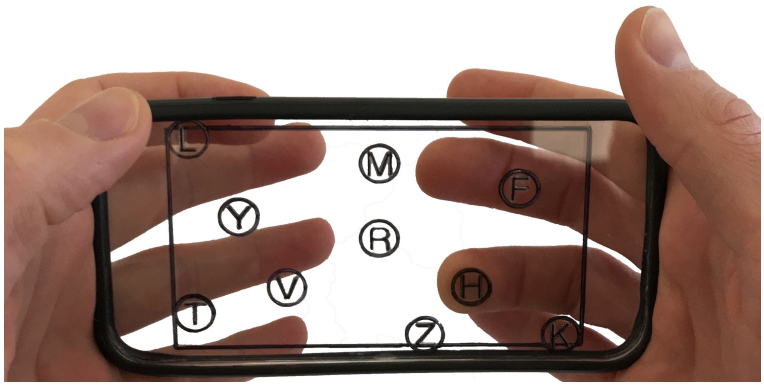


Figure 4.2: Exemplary hand and finger pose for exerting BoD force on the 4.7" prototype from Study 1. Unlike the little fingers, index, middle, and ring fingers are usually located behind the device. Both thumbs can reach the entire frontal screen represented by a black rectangle.

counts), followed by the ring finger (four counts). These results were the same for both hands and similar for the 5.5" form factor. When asked which device size users preferred for BoD force input, eight out of twelve voted for the 4.7" landscape form factor. Based on all these results, we decided to focus on a 4.7" landscape form factor for index, middle, and ring finger BoD force input for our next studies.

4.5 Study 2: BoD Force Finger Pose

In Study 2, we wanted to understand size and locations of where users place their index, middle, and ring fingers from both hands to exert BoD force when the device is held in landscape orientation.

We now wanted to understand at what areas users place their index, middle, and ring finger at the BoD to exert force. We sandwiched two iPhones to the back of each other—a common technique to read input from both, back and front [Schoenleben et al., 2013; Shen et al., 2009; Stewart et al., 2010; Wolf et al., 2012]. At the front, we used an Apple iPhone 6, whereas at the back a force-sensitive iPhone 6s was used. Force sensitivity was set to "firm", giving a range of pressure values from 0 to $\frac{480}{72} \approx 6.67$ in steps of $\frac{1}{72}$ using a linear transfer function. Apple does not provide any information about how these values translate to Newtons, but experiments [Nelson, 2015] hint at a maximum close to 4 N. Both devices share the same 138×67 mm^2 width and height

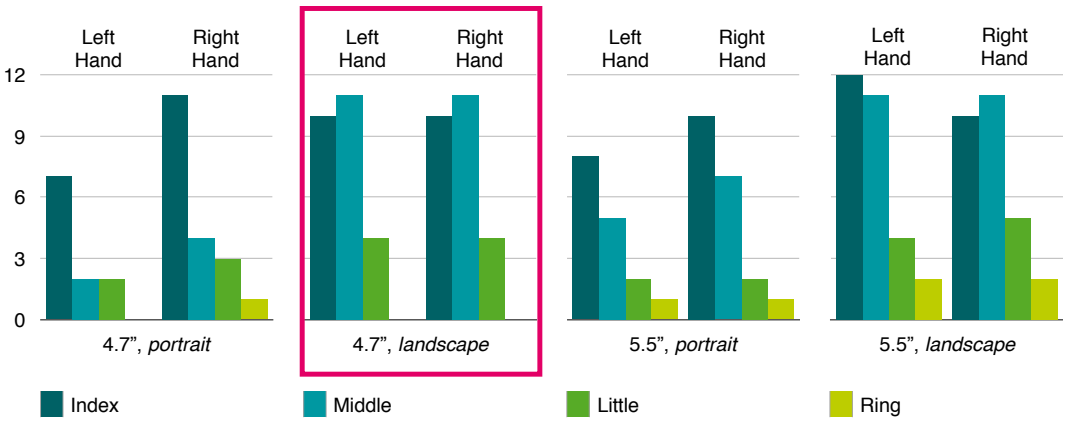


Figure 4.3: Frequency counts for the fingers used for exerting BoD force. Users preferred the small device held with two hands in landscape orientation (red box).

with a screen size of 104 × 58 mm^2, resulting in a diagonal of 4.7" at a 16:9 aspect ratio. Users were asked to hold the device in landscape orientation using the typical posture observed in Study 1. The force-sensitive side was facing away from the user.

We added a force-sensitive iPhone to the back of another iPhone and instructed participants to exert different intensities of force with each of the six fingers.

The touchscreen at the front was divided into six equally-sized areas. Each area represented a target that was to be pressed from behind using one of the six fingers that were located roughly behind the target. E.g., the target at the upper left area was to be pressed with the left index finger. We tested three different force levels: low (1.50–2.50), medium (3.84–4.84), and high (6.17–6.67). We ignored force below the lowest level to disambiguate resting fingers from force input. According to Apple's iOS SDK [Apple Inc., 2019b], a value around 1.0 represents a normal tap, whereas values beyond that level represent intentional force input. Although the range for the highest force level was half as small as for medium and low, it actually had infinite target width: users could always apply more force than the device could sense to achieve the highest level. By default, a target had a light gray color. Users had to exert force with the corresponding finger at the back until the target turned green. If users applied too much force, they left the requested force range and the target turned red; when force was too low, the target turned blue. As soon as the user stayed within the correct force level for 1 s, the next trial was shown.

Each participant did 6 × 3 × 3 (Finger × Force Level × Repetition) = 54 trials. Finger was randomized. Once all fingers were tested, a Repetition was done, and after three repetitions in total, the next Force Level was chosen. To get acquainted with the system, users performed test trials beforehand.

Each participant performed 54 force touch trials.

Overall, 12 participants (22–57 years, M = 41.83, SD = 13.09, five females, two left-handed) participated. Users' finger lengths were M = 58.33 mm (SD = 4.64) for the thumb, M = 75.42 mm (SD = 4.34) for the index finger, M = 81.75 mm (SD = 4.97) for the middle finger, and M = 76.00 mm (SD = 5.59) for the ring finger. Eight users regularly used a smartphone with a screen diagonal of about 5", and four subjects used a smartphone with a screen sized between 3.5" and 4".

12 participants took part in Study 2.

Results

Figure 4.4 shows the distribution of force touches at the back. As can be seen, touches from both hands almost never spread across the center of the x-axis, thus both sides can be distinguished in software by cutting that axis in half. One participant (aged 50, right-handed, female) tended to exert force by the left hand closer to the center of the x-axis than all other users, such that the touch ellipses for the left hand are wider than for the right hand. Regarding the spread in y-direction, force touches of adjacent fingers of each hand showed some overlap. Touches from the index finger had the smallest y-spread, whereas middle finger touches spread equally up and down from the center of the y-axis, and touches performed by the ring finger did not usually occur below 5 mm on that axis. We calculated the effective height [Soukoreff et al., 2004] for each finger area by multiplying the standard deviation of the force touches y-direction with a factor of 4.133, such that 96% of the touches would be contained within a finger area. This height was similar for the same finger types of each hand. Still, these heights had a slight overlap between adjacent fingers, such that we took the center of these overlays as separators between fingers (Fig. 4.4, gray lines). These separators serve as a heuristic to identify from which finger force is exerted at the back, which we used in our next study.

Force touches form both hands did not cross the vertical center, but there were slight horizontal overlaps between adjacent fingers from the same hand.

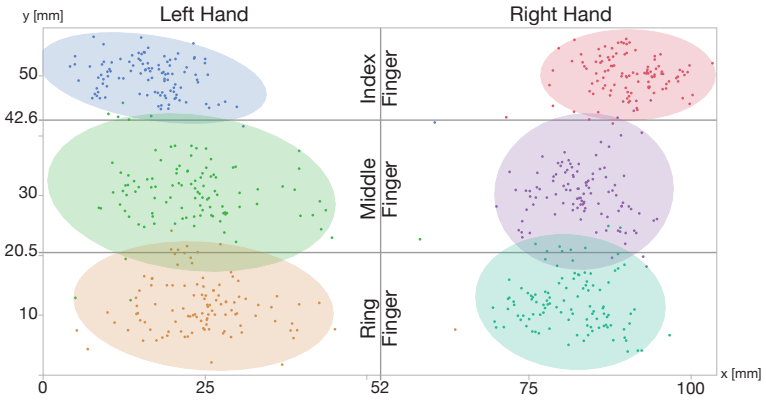


Figure 4.4: Distribution of BoD force touches from Study 2 split across hands and fingers. Ellipses denote 95% coverage. Gray bars are heuristic divides to identify the finger that performed a force touch.

4.6 Study 3: Force Dynamics and Tapping Accuracy

In Study 3, we wanted to understand how fast and accurately participants can exert BoD force while tapping one (1-Tap) or multiple (N-Tap) touch targets at the frontal touchscreen.

Our next goal was to understand the usability of BackXPress for our two use cases 1-Tap and N-Tap: 1-Tap addressed the question of *"How accurately can users apply BoD force for a single tap at the front of a smartphone while holding it with two hands in landscape orientation?"*. N-Tap addressed the question of *"How accurately can users maintain BoD force over time while tapping several targets in sequence?"*. Both tasks are inspired by McLachlan et al. [2014], who called them *Targeting* and *Maintaining*. We applied some modifications to these tasks to match the particular features of BackXPress. Upfront, we expected that BoD force would increase upon tapping, since this pushes the device against the fingers at the back, that are in a static position and hence repel that additional force. We also expected BoD force to affect frontal tapping negatively. For N-Tap, we expected the number of frontal taps to be smaller than for tapping without BoD applying force.

4.6.1 Experimental Design

Apparatus

We reused the devices from Study 2. This time, however, we improved on the perceived device thickness that our users from Study 2 found unnaturally thick. We added a 3D-printed case with stand-away sides that were just 7 mm thick (Fig. 4.5).

We reused the Study 2 apparatus but reduced the perceived device thickness at the sides.

Participants

18 participants (19–35 years, M = 25.06, SD = 4.58, seven females, three left-handed) participated in this study. Users' finger lengths were similar to those from Study 2. Thumb: 57.61 mm (SD = 5.53), index finger: 75.72 mm (SD = 5.34), middle finger: 81.06 mm (SD = 5.48), ring finger: 76.78 mm (SD = 5.45). All subjects regularly used a smartphone with a screen size of about 5". Half of them started with *1-Tap*.

18 participants took part in Study 3.

4.6.2 Task 1: 1-Tap

In 1-Tap, users were asked to apply a certain force at the BoD and then tap a touch target on the frontal touchscreen. When force was applied, visual feedback was given: A cursor highlighted the current force level on a menu visualized by a vertical color bar (Fig. 4.5). Each color represented a certain force range. Applying more force moved the cursor up, less moved it down. The color bar was displayed at the left and right on the frontal touchscreen such that it was visible even when it was hidden by the thumb tapping.

For the 1-Tap task, users were asked to exert certain BoD force with a certain finger and tap on on-screen target with one of the thumbs at the frontal touchscreen.

A white vertical line next to the force menu indicated which of the six fingers to use for force navigation: The back was split into 3 × 2 regions, one for each finger, based on the results from Study 2. If, e.g., the finger indicator appeared on the left hand side next to the the upper third of the menu, then the user had to use her index finger of the left hand. Touch targets were displayed on the frontal screen one at a time as a crosshair. The crosshair color matched the requested force range from the colored menu

The frontal touchscreen showed all task instructions to the participant.

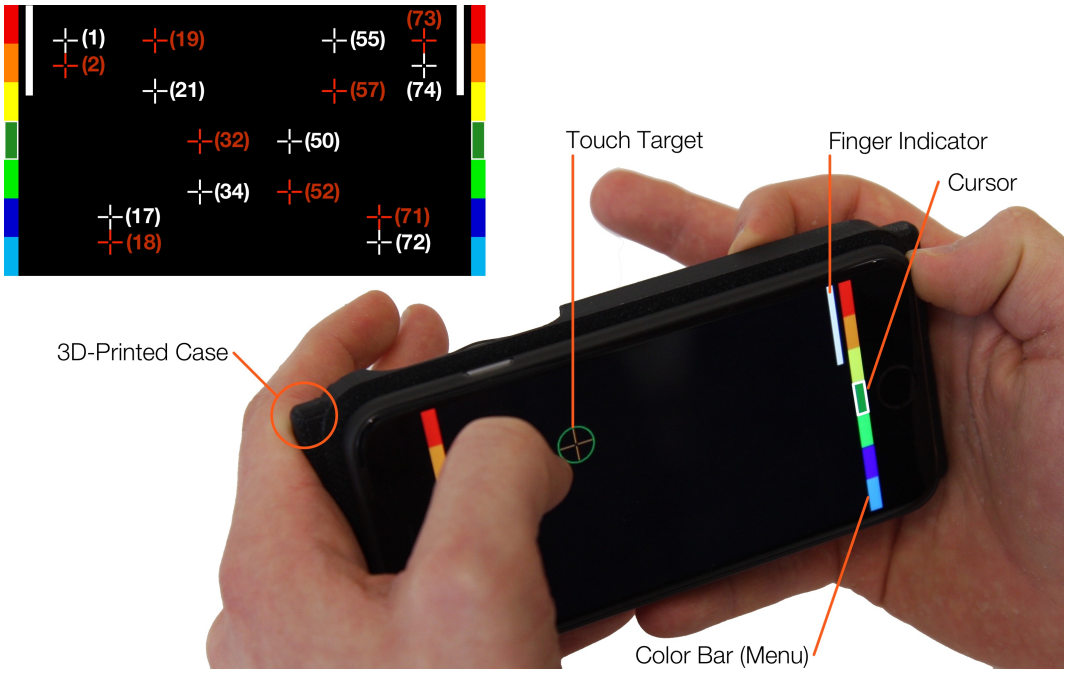


Figure 4.5: Top: Distribution of Touch Targets for Study 3. Targets were arranged on an invisible 9 × 9 grid with a 12.5% offset from the corners of a 1334 × 750 px area. White crosses show targets for 1-Tap. Red crosses are mirrored along the y-axis and were used for N-Tap. Bottom: Prototype with UI with 7 menu items for Study 3. The user has to exert BoD force with the right index finger, which indicated by the white vertical bar. Currently, the 4th item (dark green) is selected, which is indicated by both a white rectangular cursor around the item and a color-matching circle around the touch target. The touch target is shown as a cross matching the target force (here: orange).

item. A circle around the crosshair visualized the currently applied BoD force by color: A color match between crosshair and circle signaled the participant that the desired force range was reached. Tapping the touch target completed the trial: As soon as finger contact with the frontal touchscreen was detected, force sensing was locked, such that the color bar cursor would not move anymore. This *force lock* was used to facilitate force maintenance upon tapping and to avoid that force changed during selection. Upon finger release, force lock was disabled again. Users were asked to touch the crosshair as accurately as possible.

Variables

Independent variables were Menu Size, Force Level, Finger, and Touch Target.

1-Tap had four **independent variables**: Menu Size, Force Level, Finger, and Touch Target. Finger referred to the finger to be used to exert force at the back and had six levels: index, middle, and ring finger of each hand. Menu Size split the force range between 1 and 6 equidistantly into 3, 5, or 7 discrete items. We did not exploit the full force range up to ≈6.67, to avoid infinite target width, that would make controlling force easier for the last menu item by simply pressing as hard as possible. For practical application, however, an infinite target width should be considered. We also added a baseline condition to contrast the user's pure touch performance with the influence of BoD force. Hence, this condition only involved tapping at the front and ignored force input completely. Force Level had three values: 1.83, 3.50, and 5.17—these were the centers of the items for Menu Size 3. Users did not have to reach these exact values, but they should just stay within the corresponding menu item. For Menu Size 3, all three items were tested, for Menu Size 5 items 1, 3, and 5 were tested, and for Menu Size 7, items 2, 4, and 6 were tested. This way, not all items for a Menu Size were tested, but it ensured that an equal number of measurements per Menu Size was obtained—an approach that has also been applied by McLachlan et al. [2014], G. A. Ramos et al. [2007], Wilson et al. [2010], and Wilson et al. [2011]. Touch Target had eight levels. These were the targets that users had to tap at the front using their thumbs. The targets were positioned on an invisible 9 × 9 grid (Fig. 4.5, top). These locations were chosen such that users had to touch at different regions in proportion to the finger exerting BoD force, e.g., the thumb being close to that finger or far away, since such constellations could affect force maintenance differently.

We ensured counterbalancing and randomization for the presentation of the trials to each participant.

Menu Size was counterbalanced using a Latin Square. Finger, Force Level, and Touch Target were counterbalanced altogether to ensure that the user had to release force after each trial, which is typical for 1-Tap interactions. After all trials for a Menu Size were tested, the participant was given a one minute break to relax her arms and fingers. A left Touch Target was always succeeded by a right one and vice versa to balance the usage of the thumbs. Half of the participants started at the left. Users were not forced to use a particular thumb for a Touch Target, but

they usually tapped targets on each side with the corresponding thumb.

Dependent variables were the task completion *Time*, the *Success* of tapping a target with requested BoD force, the *Touch Error* as Euclidean distance to the touch target, and the *Force Range* between minimum and maximum exerted force in a trial.

The **dependent variables** were the *Time* (ms) between a target appeared and the user releasing her finger from it; *Success* (yes/no) whether the touch target was selected with the correct item from the force menu; *Touch Error* (mm) as Euclidean distance between the center of a given touch target and the user's touch at the front; and *Force Range* (relative, on a normalized scale from 0–1) between the minimum and maximum BoD force values sensed between the user touching and releasing the frontal touchscreen. *Success* and *Force Range* were not measured for the baseline condition, since it did not involve any force measurement. Likewise, Finger and Force Level were inexistent in the baseline condition, since these variables referred to BoD force execution.

Including baseline trials that involved tapping without BoD force, each participant performed 456 trials.

1-Tap had 3 × 6 × 3 × 8 (Menu Size × Finger × Force Level × Touch Target) = 432 force trials. For comparison, users performed 3 × 8 = 24 baseline trials, hence 1-Tap had 456 trials per participant in total. A trial ended as soon as the user lifted the thumb off the touchscreen, independent from whether force was within the correct range for Force Level or how accurately the Touch Target was hit. Before data was recorded, users had 24 test trials.

4.6.3 Results – 1-Tap

We first present the results for data compatible with the baseline condition, followed by data from BoD force conditions.

Baseline Data

As to be expected, *Time* increased with Menu Size.

Time. We conducted a two-way repeated measures ANOVA on the log-transformed *Time* data. There was a significant main effect for Menu Size ($F_{3,8156} = 245.05$, $p < .0001$). Tukey-HSD post hoc pairwise comparisons revealed significant differences across all Menu Sizes ($p < .01$, each). *Time* increased with the number of menu items (Fig. 4.6, left). There was no main effect for Touch Target and there were no interaction effects.

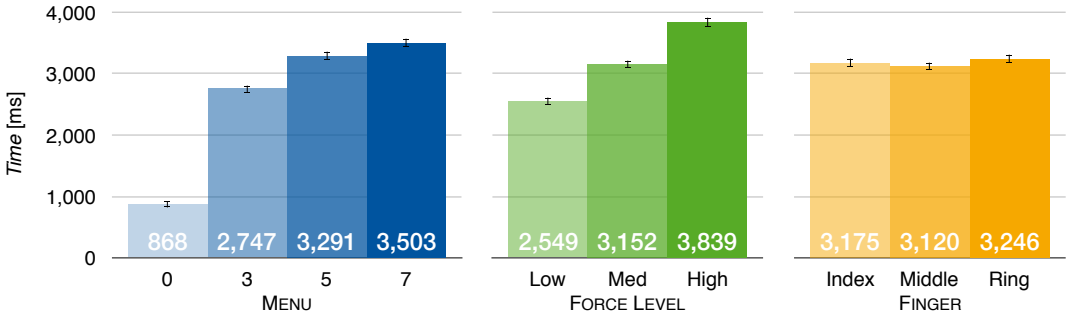


Figure 4.6: *Time* [ms] data for 1-Tap split by Menu Size, Force Level, and Finger. Menu Size 0 was the baseline condition. Error bars denote 95% CI.

Touch Error. A two-way repeated measures ANOVA on *Touch Error* showed a significant main effect for Touch Target ($F_{7,8156}$ = 70.45, p < .0001). Tukey-HSD post hoc pairwise comparisons revealed that *Touch Error* for targets 1, 17, 21, and 74 was significantly lower compared to all other targets (p < .05, each) (Fig. 4.7). However, the difference of up to 1.10 mm between these two target groups is relatively small and practically negligible. There was no main effect for Menu Size and no interaction effect, indicating that BoD force had no influence on frontal tapping accuracy. On average, *Touch Error* was 2.47 mm (95% CI: [2.44; 2.50] mm).

On average, *Touch Error* was 2.47 mm.

BoD Force Data

Time. We ran a four-way repeated measures ANOVA on the log-transformed *Time* data. Apart from the significant main effect for Menu Size (cf. baseline analysis), there was a significant main effect for Force Level ($F_{2,7540}$ = 160.77, p < .0001). Tukey-HSD post hoc pairwise comparisons showed significant differences across all levels (p < .0001, each), increasing from lowest to highest (Fig. 4.6, middle). This was to be expected since users navigated pressure upwards starting from zero, hence higher force levels were further away and required more time. There was also a significant main effect for Finger $F_{2,7540}$ = 6.57, p < .01). Tukey-HSD post hoc pairwise comparisons revealed that exerting BoD force with the ringer fingers took significantly more time compared to index and middle fingers (p < .05, both) (Fig. 4.6, right). There was also a Menu Size × Finger interaction effect ($F_{4,7540}$ = 2.64, p < .05), that confirmed that users con-

As to be expected, *Time* increased with Force Level.

		Menu Size				Force Level			Finger		
		Baseline	3	5	7	Low	Medium	High	Index	Middle	Ring
Touch Error [mm]	M	2.19	2.42	2.48	2.55	2.52	2.47	2.47	2.50	2.49	2.46
	SD	1.15	1.24	1.23	1.31	1.27	1.27	1.24	1.31	1.24	1.24
Force Range [%]	M	N/A	6.29	5.76	7.27	5.77	6.46	7.09	7.22	6.29	5.82
	SD	N/A	9.94	9.13	12.86	12.36	10.65	9.04	11.30	10.85	10.11

Figure 4.7: *Touch Error* [mm] and *Force Range* [%] data for 1-Tap split by Menu Size, Force Level, Finger, and Time Block.

trolled BoD force significantly faster for a combination of fewer menu items with lower force values compared to more menu items paired with higher force values.

Touch Error. We ran a four-way repeated measures ANOVA on *Touch Error*. Apart from the significant main effect for Touch Target (cf. baseline analysis), there were no further main effects or interaction effects.

Using three menu items, low force, and the middle finger led to the highest *Success*.

Success. We ran Cochran's Q tests on the dichotomous *Success* data. There was a significant main effect for Menu Size ($Q(2) = 232.66$, $p < .0001$), Force Level ($Q(2) = 75.89$, $p < .0001$), Finger ($Q(2) = 29.38$, $p < .0001$), and Touch Target ($Q(7) = 30.44$, $p < .0001$). Post hoc pairwise comparisons for Menu Size were all significant ($p < .0001$, each), with Menu Size 3 leading to the highest *Success* (86.8%, Fig. 4.8, left). Post hoc pairwise comparisons for Force Level showed that the highest level led to significantly lower *Success* compared to the medium and lowest level ($p < .0001$, both) (Fig. 4.8, middle). Post hoc pairwise comparisons for Finger revealed that BoD force exertion by middle finger led to significantly higher *Success* compared to index and ring finger ($p < .0001$, both) (Fig. 4.8, right). Post hoc pairwise comparisons for Touch Target showed that outer targets 1, 72, and 74 led to significantly higher *Success* (79.9–82.3%) compared to center target 34 (73.7%) ($p < .0001$, each). Interestingly, *Success* for center target 50 (76.9%), was not significantly different from outer targets.

On average, *Force Range* was 6.44%.

Force Range. A four-way repeated measures ANOVA on the normalized *Force Range* data showed a significant main effect for Finger ($F_{2,7540} = 7.48$, $p < .001$) and Touch Target ($F_{7,7540} = 3.99$, $p < .001$). Tukey-HSD post hoc pairwise comparisons between

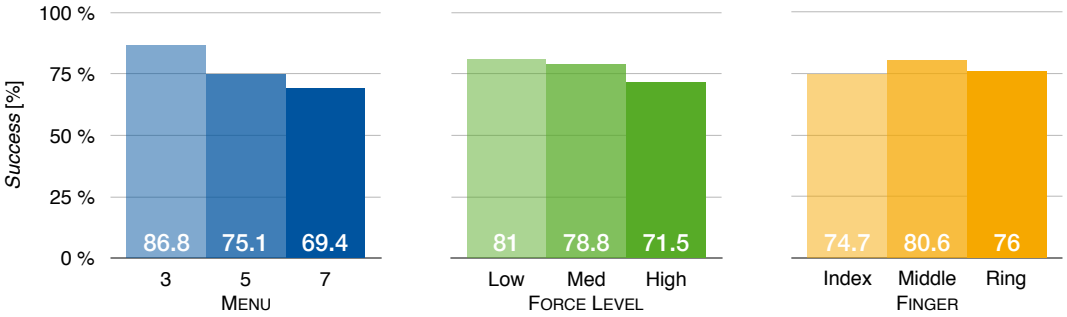


Figure 4.8: *Success* [%] for 1-Tap split by MENU SIZE, FORCE LEVEL, and FINGER.

different FINGERS showed that *Force Range* for the index finger was significantly higher compared to middle and ring finger ($p < .05$, both) (Fig. 4.7). However, the difference of ≈1.00% is practically negligible. For TOUCH TARGET, Tukey-HSD post hoc tests revealed that *Force Range* for targets 21 and 72 was significantly higher compared to targets 74 and 1 ($p < .05$, each), but, again, at a difference of ≈1.00% practically negligible. On average, *Force Range* was 6.44% (95% CI: [6.20; 6.68]%).

4.6.4 Discussion – 1-Tap

Success decreased with MENU SIZE and was almost 90% when the middle fingers were used to select from three menu items.

Not surprisingly, *Success* decreased with MENU SIZE. BoD force control with selection at the front worked best for three menu items (86.6%). When only the middle finger was used, a success rate of 89.5% was achieved. We believe that the middle fingers performed best, because they are located at the horizontal center of the device, where applying force hardly tilts the device towards the user. Furthermore, users were more successful when controlling low force. Overall, *Success* was fairly acceptable, but bimanual touch input and force control at the front of a tablet reached the same *Success* for 10 menu items [McLachlan et al., 2014]. We are confident, however, that potential learning effects and using an infinite target width for the last item of each MENU SIZE will lead to a higher *Success* since then the user could just press as hard as possible to reliably reach the last menu item.

Exerting high forces was considered annoying by most participants because

Time increased with MENU SIZE even though the same FORCE LEVELS had to be acquired. Some users stated that force control for MENU SIZES 5 and 7 "[...] was a bit annoying" since the cursor was jumping a lot. Consequently, participants needed more time

this takes significant time.

to carefully control force. Reaching for highest Force Level was also considered negatively: "It takes longer to reach high force.". Not surprisingly, BackXPress was slower than plain touching at the front (baseline condition), since the timings for BackXPress included navigation time for the menu. Still, BackXPress was slower than McLachlan et al. [2014], where users controlled force for seven to ten items at the front of a tablet within the same time. Hence, we conclude that force navigation at the BoD is more demanding than at the front.

Tapping at the front led to an involuntary increase in maintained BoD force.

Upon tapping, BoD force increased, but it dropped back to almost the same value as soon as the thumb was lifted off the front. Overall, this change in force was small (6.29% for Menu Size 3, cf. Fig. 4.7), which is about half the force applied during a normal tap on a touchscreen. Still, it makes sense to apply the force lock technique, because when force is applied near a force range boundary, the cursor might still jump to the adjacent bar, confusing the user. *Touch Error* was not affected by BoD force, which is good and in-line with previous findings from McLachlan et al. [2014].

Overall, for interactions like 1-Tap, we recommend a Menu Size of three items and using the force lock technique. If only up to six different modes are needed, let users use their middle fingers to enter those modes.

4.6.5 Task 2: N-Tap

For the N-Tap task, participants had to maintain certain BoD force for 20 s while tapping as many targets as possible at the frontal touchscreen.

N-Tap was similar to 1-Tap, but users had to maintain Force Level for 20 seconds while tapping Touch Targets. Users were instructed to tap as many Touch Targets as possible, but not at the cost of maintaining the correct Force Level or tapping accuracy. As in 1-Tap, force lock was only active per tap and not the entire 20 seconds, since, in reality, the device would not know whether the user intends to tap only once or a sequence of Touch Targets for a fixed Force Level. We used the same UI as for 1-Tap to visualize the task. Each trial ended after 20 seconds.

Variables

The **independent variables** Menu Size, Force Level, and Finger were the same as for 1-Tap. Again, we added a baseline condition that involved just tapping at the front for 20 seconds. In addition, Time Block assigned users' taps on touch targets to one of the four equally-sized time ranges: 0–5, 5–10, 10–15, and 15–20 seconds, based on the time when the device registered first contact with the touchscreen. This was done to analyze how users' performance developed over time. In contrast to 1-Tap, Touch Targets were mirrored along the vertical axis (Fig. 4.5, top) to avoid learning effects. Menu Size was counterbalanced and Finger and Force Level were counterbalanced together using a Latin Square. The participant was given a one minute break to relax her arms and fingers until the next Menu Size was chosen. Touch Target was completely randomized, but left and right targets always alternated. Half of the participants started at the left.

In addition to the independent variables used for the 1-Tap task, we added Time Block to split analysis of the 20 s of tapping into four equal time chunks of five seconds.

Dependent variables were *Touch Error* and *Force Range* as defined for 1-Tap. In addition, *Tap Count* measured how many touch targets users tapped within the 20 seconds. We did not investigate *Time*, as each trial lasted exactly 20 seconds. Again, *Success* and *Force Range* were not measured for the baseline condition, and Finger and Force Level did not exist for this condition.

Tap Count counted how many frontal taps the participant achieved within the 20 s of maintaining BoD force.

In total, N-Tap had 3 × 6 × 3 (Menu Size × Finger × Force Level) = 54 force trials. For comparison, users performed three baseline trials, resulting in 57 trials per participant in total, each lasting 20 seconds. Users had three test trials to get familiar with the task before data recording started.

Each participant performed 57 trials.

4.6.6 Results – N-Tap

Again, we first present the results for baseline condition compatible data, followed by data from BoD force conditions.

Baseline

Touch Error was 2.24 mm, thus similar to results from the 1-Tap task.

Touch Error. A three-way repeated measures ANOVA revealed a significant main effect for Touch Target ($F_{7,11602}$ = 14.00, p < .0001). Post hoc Tukey-HSD pairwise comparisons showed that upper targets 2, 19 and 73 led to a significant lower *Touch Error* compared to center targets 52, 32, and 57 (p < .01, each). However, the difference between these two target groups of up to 1.00 mm is relatively small and practically negligible. There were no significant main effects for Menu Size and Time Block and there were no interaction effects. On average, *Touch Error* was 2.24 mm (95% CI: [2.22; 2.26] mm) (Fig. 4.9).

Tap Count decreased with increasing Menu Size and was lowest within the last five seconds of maintaining BoD force.

Tap Count. A Friedman test showed a significant main effect for Menu Size ($\chi^2(3)$ = 157.92, p < .0001). *Tap Count* decreased with increasing Menu Size (Fig. 4.10, left). Post hoc pairwise comparisons were all significant (p < .0001, each) except between Menu Sizes 5 and 7. There was also a significant main effect for Time Block ($\chi^2(3)$ = 1376.47, p < .0001). Post hoc pairwise comparisons were all significant except between Time Blocks 5–10 s and 10–15 s. *Tap Count* was lowest within the first five seconds, then increased and stabilized from five to 15 seconds, but slightly decreased afterwards, possible due to fatigue. There was also a Menu Size × Time Block interaction effect ($\chi^2(9)$ = 2.76, p < .005). Any Time Block for Menu Size 0 was faster than any Time Block from Menu Sizes 3, 5, and 7. Furthermore, for these Menu Sizes, *Tap Count* was similar and lowest within the first five seconds (Fig. 4.10, right).

BoD Force Data

Touch Error. We ran a five-way repeated measures ANOVA on *Touch Error*. Apart from the significant main effect for Touch Target (cf. baseline analysis), there were no further main effects or interaction effects.

Tap Count decreased with increasing Force Level.

Tap Count. In addition to the significant main effects for Menu Size and Time Block (cf. baseline analysis), a Friedman test showed a significant main effect for Force Level ($\chi^2(2)$ = 618.06, p < .0001). *Tap Count* decreased with increasing Force Level (Fig. 4.10, middle).

		Menu Size				Force Level			Finger			Time Block [s]			
		Baseline	3	5	7	Low	Medium	High	Index	Middle	Ring	0–5	5–10	10–15	15–20
Touch Error [mm]	M	2.26	2.20	2.24	2.28	2.23	2.26	2.22	2.26	2.22	2.23	2.18	2.27	2.28	2.19
	SD	1.99	1.19	1.14	1.19	1.18	1.12	1.22	1.19	1.21	1.12	1.11	1.26	1.17	1.10
Force Range [%]	M	N/A	2.04	1.76	2.36	1.50	2.16	2.70	2.24	1.86	2.04	2.60	1.89	1.85	2.10
	SD	N/A	4.15	3.60	5.20	4.03	4.59	4.29	4.14	4.22	4.58	5.25	4.07	3.92	4.39

Figure 4.9: *Touch Error* [mm] and *Force Range* [%] data for N-Tap split by Menu Size, Force Level, Finger, and Time Block.

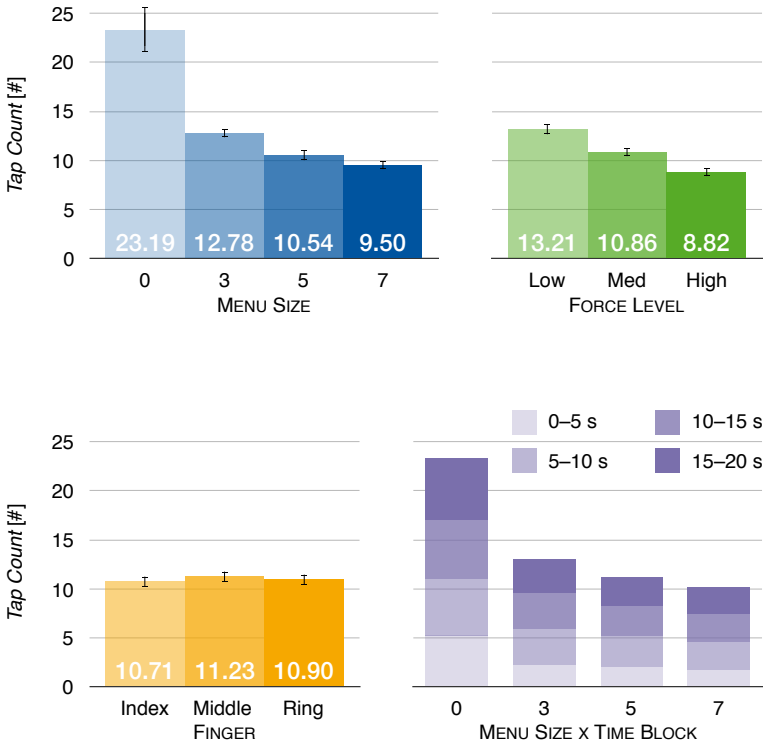


Figure 4.10: *Tap Count* for N-Tap split by Menu Size, Force Level, Finger, and Menu Size × Time Block. Error bars denote 95% CI.

Post hoc pairwise comparisons were all significant ($p < .0001$, each).

Success. We ran a Cochran's Q test on the dichotomous *Success* data. There was a significant main effect for Menu Size ($Q(2) = 181.81$), Force Level ($Q(2) = 117.03$), Finger

Success was lowest withing the first five seconds, indicating that participants needed to get familiar with the task.

($Q(2) = 17.33$), and Time Block ($Q(3) = 64.47$), $p < .0001$, each. *Success* decreased with increasing Menu Size (Fig. 4.11, left). Post hoc pairwise comparisons for Menu Size were significant between Menu Sizes 3 and 5 and between 3 and 7 ($p < .0001$, each), as well as between 5 and 7 ($p < .05$). Pairwise comparisons for Force Level were all significant ($p < .05$, each). *Success* decreased with increasing Force Level (Fig. 4.11, middle) indicating that higher BoD force was more difficult to maintain. Pairwise comparisons for Finger showed that BoD force exerted by the middle finger led to significantly higher *Success* compared to ring and index finger ($p < .05$, each) (Fig. 4.11, middle). Regarding Time Block, *Success* was significantly lower within the first five seconds compared to the remaining time slots ($p < .0001$, each), indicating that users needed some time to get familiar with pressure navigation (Fig. 4.11, right). There was also a significant main effect for Touch Target ($Q(7) = 45.51$, $p < .0001$), but without a clear pattern.

On average, *Force Range* was 2.04%.

Force Range. A five-way repeated measures ANOVA on the normalized *Force Range* data showed a significant main effect for Force Level ($F_{2,10305} = 27.72$, $p < .0001$) and Time Block ($F_{3,10380} = 21.04$, $p < .0001$). Tukey-HSD pairwise comparisons revealed that *Force Range* for the highest Force Level was significantly higher compared to the medium and low levels ($p < .0001$, both) (Fig. 4.9). However, the difference was only 1–2% and therefore practically negligible. Tukey-HSD pairwise comparisons between different Time Blocks were significant between 0–5 seconds and all other blocks ($p < .005$, each), but the difference was below 1% (Fig. 4.9). There was also an interaction effect for Menu Size × Finger ($F_{4,10305} = 2.53$, $p < .05$), but post hoc pairwise comparisons were not significant. On average, *Force Range* was 2.04% (95% CI: [1.96; 2.13]%).

4.6.7 Discussion – N-Tap

While overall, the results were similar to those from the 1-Tap task, participants needed relatively more time to complete the

Like for 1-Tap, *Success* dropped with Menu Size, and reached up to 89.1% for Menu Size 3 (Fig.4.11, left). *Success* was about 5% lower within the first five seconds compared to the remaining 15 seconds, suggesting that users were getting familiar with the task and that with regular use, results would be the same across the entire interaction time of 20 seconds. As for 1-Tap, BoD force

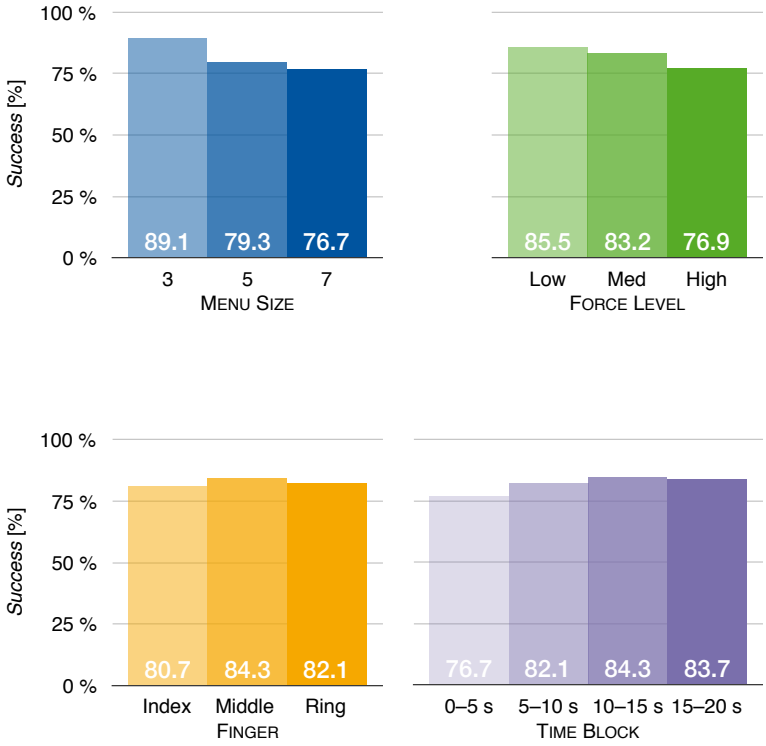


Figure 4.11: *Success* [%] for N-Tap split by Menu Size, Force Level, Finger, and Time Block.

had no impact on *Touch Error*. Maintaining BoD force, however, slowed down the interaction significantly, far more than we expected. Ideally, the user would acquire the desired force level once and then consecutively touch at the front, which users can do quickly, as the baseline condition showed. This, however, would not lead to a drop of about 50% in frontal taps for Menu Size 3 compared to the baseline condition. This could mean that users needed to re-adjust the cursor after tapping—especially for the highest Force Level—or they were more careful to not change their force, which would slow down the interaction for both cases.

trials, potentially due to a required correction of the BoD force that could drop during the 20 s because of fatigue or the influence of frontal tapping.

Basically, we could not compare our N-Tap results with those from McLachlan et al. [2014], since our measurements and Force Levels were different. Only our medium Force Level compares to the 2 N level from McLachlan et al. [2014]. Here, users deviated by 0.15 N (*SD* = 0.12 N), whereas with BackXPress, users deviated by 0.22 N (*SD* = 0.20 N) on average, hence, more. This

Maintaining force at the BoD is more difficult than at the front.

indicates, again, that controlling and maintaining force at the BoD is more difficult than at the front.

Using an infinite target width for the last item of each menu is likely to increase *Success* for both, 1-Tap and N-Tap tasks.

Overall, performance for N-Tap was similar to the results from 1-Tap. Therefore, we give the same recommendations for both tasks, i.e., using a Menu Size of three items, with preferring the use of the middle fingers. However, in both tasks, we intentionally did not use an infinite target width for the last item of each Menu Size. In practice, using an infinitive target width, however, will likely increase *Success* for both 1-Tap and N-Tap and also *Tap Count*, since the user could then press as hard as possible to quickly reach the last menu item.

4.7 Improving Dynamic Force Control

Instead of confirming BoD force via a single time stamp from the frontal tap event, using an average of the BoD force values captured 500 ms before the tap event occurs significantly improves users' *Success*.

For both 1-Tap and N-Tap, force control and selection follow a clear sequential pattern: The user first applies BoD force to enter the desired mode, then selects a target on the touchscreen. We were interested in seeing how force changed between these two events. Therefore, we looked into our data streams that continuously sampled BoD force. To compare force curves across different taps, we set the time at which the thumb started hitting the touchscreen, denoted as *Contact Event*, as zero. Figure 4.12 shows an example. As can be seen, force stabilized up to 750 ms before a Contact Event, which we interpreted as that the user had controlled to the desired menu item and was about to touch the screen. Note that the force values were towards the lower end on the correct force range. This is likely due to the fact that users navigated force from below and were about to tap as soon as the UI indicated that the correct item was reached. For each Contact Event, we calculated the mean pressure from t ms before the event until 0 ms, the time of contact. Although in Figure 4.12 the force exceeded the force range about 100 ms before the tap, averaging until 0 ms lead to better results because it helped pushing the average force a little bit up, compared to the values before 100 ms, which were, as stated above, at the lower end of the force range. Figure 4.13 shows a typical force curve for N-Tap. Using this averaged force upon touch target selection, we could increase *Success* across all Menu Sizes, with a value of $t = -500$ ms for 1-Tap and a value of $t = -700$ ms for N-Tap leading to the best results. Pairwise McNemar tests be-

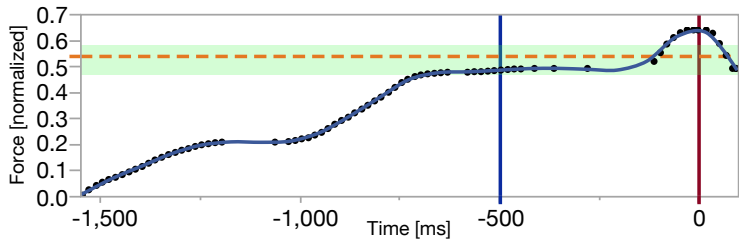


Figure 4.12: Force curve for a single trial for *1-Tap* from one user using the ring FINGER. The green area shows the force range for the medium FORCE LEVEL for MENU SIZE 7. At the time the user tapped at the front (red line), she exceeded that range. Averaging force over 500 ms (blue line) before the tap, led to a better estimate (orange line) that lay within the force range.

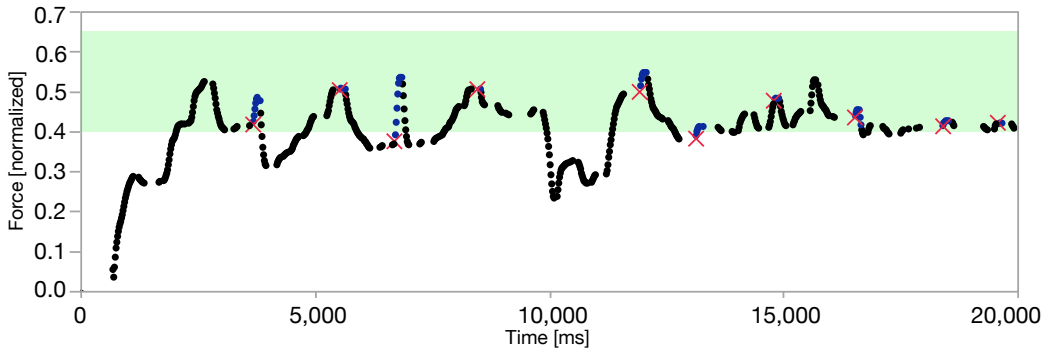


Figure 4.13: Typical 20 s force curve for N-Tap from one user using the middle FINGER. The green area shows the force range for the medium FORCE LEVEL for MENU SIZE 3. Each red cross indicates when the thumb hit the screen. Blue dots indicate that the finger was still in contact with the touchscreen. The user's force fluctuated over time, but mostly stayed withing the force range.

tween the same MENU SIZES for 1-Tap and N-Tap confirmed that averaged force lead to significantly higher *Success* compared to the original values from our study ($p < .0001$, each). This way, *Success* reached 92.4% for N-Tap with MENU SIZE 3 and increased by about 10% for MENU SIZES 5 and 7 (Fig. 4.14). Using a parameter of $t = -500$ ms for N-Tap showed no significant difference for *Success* compared to when choosing $t = -700$ ms. Hence, a combined value of $t = -500$ ms will lead to a significant increase in success for both 1-Tap and N-Tap.

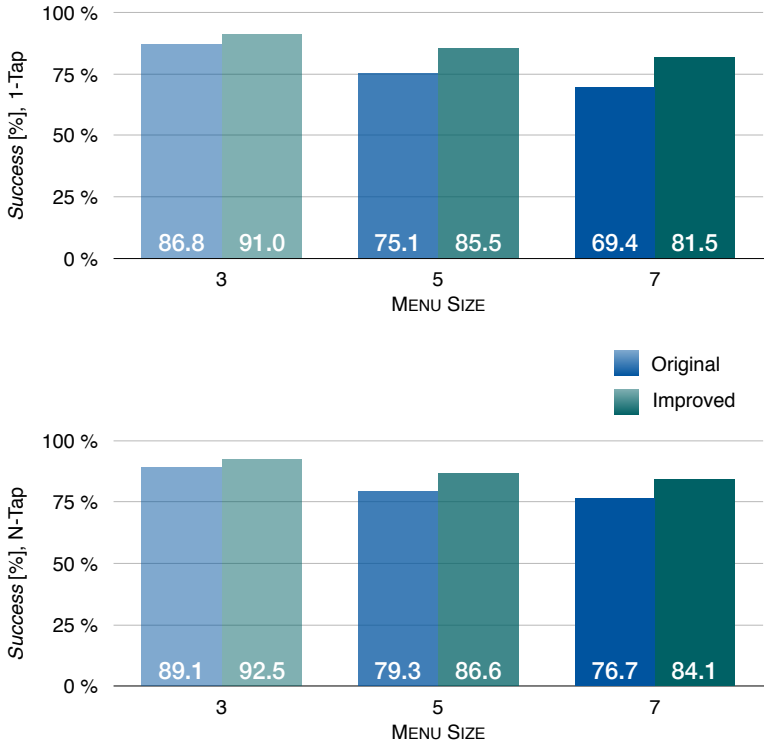


Figure 4.14: Comparison of original vs. improved *Success* [%] by Menu Size for 1-Tap (top) and N-Tap (bottom).

4.8 Design Guidelines

Based on our results, we give the following design guidelines:

1. Use a landscape-oriented device that allows both thumbs to reach the entire frontal touchscreen. Expect up to three fingers (index, middle, and ring finger) from both hands for BoD force control (Study 1). Use the force sensing area for each finger according to Figure 4.4 (Study 2).

2. Use *Force Lock*: When users touch at the front, BoD force involuntarily increases. Since the user has controlled force *before* selecting a touch target, stop force sensing while the finger is in contact with the screen. Release *Force Lock* as soon as the thumb is lifted (Study 3).

3. Use a 500 ms force history to increase success: Each time when the user's thumb hits the touchscreen to perform a selection, use an averaged force value over the last 500 ms instead of a single force value (Study 3).
4. Use three, or if absolutely necessary, five force levels above normal resting force: We obtained a 92% success for three menu items, and 87% for five items (Study 3).
5. Prefer middle fingers: If only a few modes are needed, let users control force with the middle fingers since they do this more accurately compared to the index or ring fingers. Do not consider using the little fingers (Study 3).

4.9 Limitations and Future Work

Based on our study results, we built a custom hardware prototype consisting of six FSRs, a micro controller, and a Bluetooth Low Energy chip that brings the BackXPress interaction technique to any handheld device as clip-on protection case.

We used a 4.7" sandwiched prototype to measure both touch input at the front and force input at the back. While this is a common approach applied in HCI research, both device thickness (19 mm) and weight (308 g) are higher than we would expect from a future commercial device. We reduced the perceived device thickness using a 3D-printed case (Fig. 4.5), but this added about 7 mm in width to each device side. Figure 4.15 shows a hardware prototype that we specifically designed for sensing BoD force for the six fingers used in BackXPress. We added six force sensing resistors (FSRs) to the back of a smartphone protection case that is wrapped around the device. A Bluetooth-enabled microcontroller continuously senses all force readings from each FSR and transmits the data to the smartphone application that implements the BackXPress interaction technique. Using this prototype, the overall device thickness is reduced to about 12 mm. Furthermore, the force sensor of our prototype was limited to a range of 0–4 N. However, most HCI studies about force input used sensors with a similar range. Finally, we only investigated frontal tapping but no thumb movement, e.g., for dragging, while maintaining BoD force. Using *Force Lock*, however, we expect no difference to 1-Tap: As soon as the thumb touches the screen, force changes are ignored. Only upon release, the current force likely mismatches the locked force. We will investigate this in future work. In addition, we would like to investigate simultaneous multi-finger BoD force input to combine two or more "quasi-modes", as in our hotel finding applica-

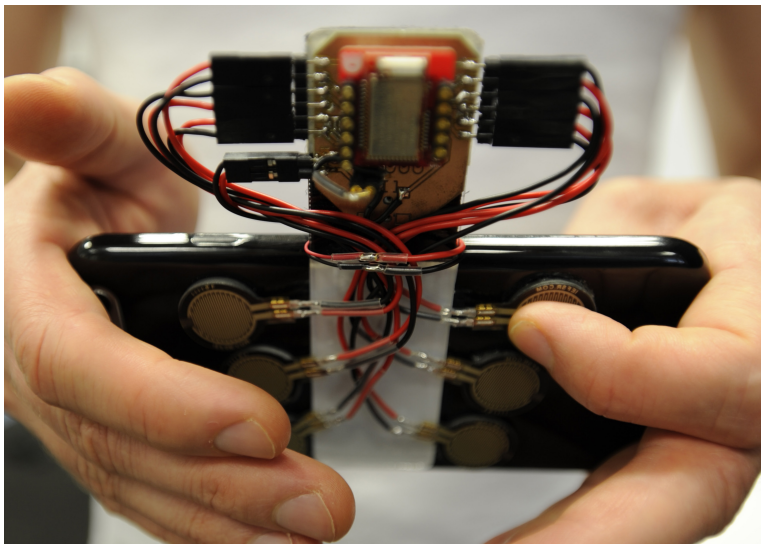


Figure 4.15: A hardware prototype specifically designed for BackXPress. Six force sensing resistors (FSRs) are glued to the back of a smartphone protection case. All FSRs are connected to a Bluetooth-enabled microcontroller that sends the individual force readings wirelessly to the smartphone. Using this prototype, we can omit the sandwiched smartphone approach from Study 3 such that the thickness of the device shrinks to 12 mm.

tion, and we would like to compare BackXPress to other existing quasi-mode techniques.

4.10 Conclusion

We designed an investigated a new interaction technique, called BackXPress, that lets users use their fingers resting at the back of their handheld device to augment the interaction with the frontal touchscreen.

We designed a new interaction technique, called *BackXPress*, that lets users create back-of-device (BoD) force input to augment their two-handed interaction on touchscreens of landscape-oriented smartphones. In Study 1 and 2 we learned that index, middle, and ring finger of both hands can reliably apply force at the BoD. With BackXPress, users can apply various force levels with each of these fingers to enter different "quasi-modes" that are only active as long as that force is applied. The thumbs of both hands then interact with the frontal touchscreen in that mode. We provided application scenarios, such as multi-language text entry, that benefit from BackXPress. In Study 3

we tested the practicability of applying BoD force during single touches at the front and frontal touch input for 20 seconds. Using a 500 ms history of averaged sampled force values *before* the user touches the screen led to a significant improvement in users' force selection. With three force levels above normal resting force, force accuracy was more than 92%. BoD force did not affect tapping accuracy at the front. We concluded with design guidelines for our interaction technique.

With BackXPress, we not only introduced a new interaction technique that augments touch input with force input, we also presented a different confirmation technique for force input. While one finger applies force to preselect a menu item, another finger, i.e., the thumb, taps to confirm the input. This confirmation technique has also been used by McLachlan et al. [2014] for force-augmented touch input on a tablet device. However, in many cases, users want to interact with handheld devices using a single hand, which leaves only the thumb available for touch and force input. While the Quick Release mechanism (cf. Section 3.1.2), by design, is an efficient confirmation technique that works with a single finger, it is practically difficult to detect reliably. Therefore, we set out to challenge this confirmation technique and present a reliable Quick Release mechanism based on user data that correctly confirms force input with 97% accuracy.

BackXPress splits force control and confirmation across two fingers. Next, we will look at a confirmation technique that allows for force control and confirmation with a single finger, e.g., for situations in which the user wants to hold the handheld device with a single hand.

5

Designing an Efficient Confirmation Technique for Handheld Force Touch Input

→ Summary

Modern smartphones, like iPhone Xs, feature touchscreens with co-located force sensing. This makes touch input more expressive, e.g., by enabling single-finger continuous zooming when coupling zoom levels to force intensity. Often, however, the user wants to select and confirm a particular force value, say, to lock a certain zoom level. The most common confirmation techniques are *Dwell Time* (DT) and *Quick Release* (QR). While DT has shown to be reliable, it slows the interaction, as the user must typically wait for 1 s before her selection is confirmed. Conversely, QR is fast but reported to be less reliable, although no reference reports how to actually detect and implement it. We set out to challenge the low reliability of QR: We collected user data to (1) report how it can be implemented and (2) show that it is as reliable as DT (97.6% vs. 97.2% success). Since QR was also the faster technique and more preferred by users, we recommend it over DT for force confirmation on modern smartphones.

Publications: The work presented in this chapter has been done in collaboration with Simon Voelker and Jan Borchers. The author of this thesis developed the research idea and relevant research questions, including the motivation of the work. Furthermore, he has designed and implemented all experiments and analyzed their data. Most of this work has been published as paper in the Proceedings of ACM ISS 2017 [Corsten et al., 2017b]. The author of this thesis is the main author of the paper. Most sections in this chapter are taken from the paper publication.

5.1 Motivation

Modern smartphones are equipped with touchscreens that can capture the force applied to each touch point.

In the previously presented interaction technique, *BackXPress*, users confirm force input exerted at the back of the device by tapping with the thumb at the frontal touchscreen. This technique clearly separates the user's control of force control from its confirmation by splitting the two across different sensors and fingers. Modern smartphones like iPhone Xs nowadays feature a touchscreen with co-located force sensors at the front to capture the normal force applied to the frontal screen with each touch. This can benefit one-handed smartphone use, for example, to enable single-finger zooming in a map at the user's finger location by coupling the continuously sampled force to the zooming level: Pressing harder zooms in, releasing the force zooms out, again. However, in many cases, force is used to select and confirm one particular value out of a given range. A recurring example from literature is menu control (e.g., Corsten et al. [2017a], McLachlan et al. [2014], G. Ramos et al. [2004], and Wilson et al. [2010]). Each discrete menu item is mapped to a different range of adjacent force values. Depending on how much force the user applies, the appropriate item is selected.

Since force is transient, selecting a particular value via force touch input requires the user to explicitly perform a confirmation gesture. One convenient confirmation technique is *Quick Release (QR)*, for which the user quickly lifts off her finger off the force sensor. However, QR has typical error rates ≥ 30%.

The key problem, however, is to confirm the selection of an item, especially when only one finger is available for both, exerting the force and confirming it. The most common techniques in the literature are *Dwell Time (DT)* and *Quick Release (QR)* [G. Ramos et al., 2004]. Using DT, the selection is confirmed after maintaining force for a short duration (usually 1 s) for the corresponding item. Using QR, the selection is confirmed by quickly lifting the finger when the item is selected (Fig. 5.1). Although DT has shown to be a reliable force confirmation technique with ≈ 97% success rates (e.g., G. Ramos et al. [2004], Stewart et al. [2010], and Wilson et al. [2011]), it slows the interaction, as the user must typically wait before her selection is confirmed. In contrast, QR is a fast force confirmation technique, but it is less reliable than DT (e.g., ≥ 30% error rates in [Cechanowicz et al., 2007; Wilson et al., 2010]). Surprisingly, literature does *not* describe how QR is actually detected and implemented [G. Ramos et al., 2004; Wilson et al., 2010].

We challenged the low reliability of QR by

Therefore, we challenged the low reliability of QR. We asked users to selected menu items by applying force with the thumb

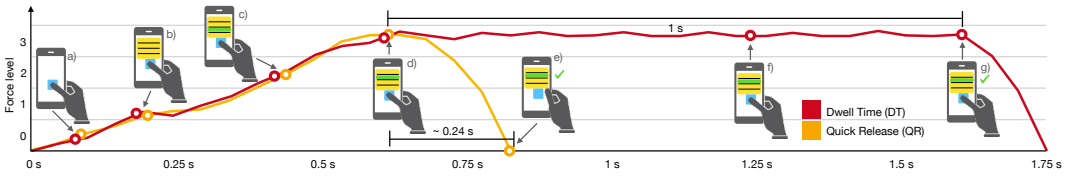


Figure 5.1: Interaction stages of force confirmation using Quick Release (QR) and Dwell Time (DT): In force level 0 (a) the user can normally interact with the system. By entering force level 1 (b), an on-screen menu is shown. Users can select a menu item by increasing the force to force level 2 (c) and force level 3 (d). Using QR, the selection is confirmed by quickly lifting the finger (e). Using DT, users have to maintain force (f) for a one second to confirm the selection (g).

and to confirm each selection by quickly lifting the thumb off the handheld touchscreen. Based on this data we present how to implement QR. In a second study, we showed that users performed with this QR implementation as accurate as with DT (97.6% vs. 97.2% success). Since QR was also the faster force confirmation technique and more preferred by users, we recommended it over DT for force input confirmation on modern smartphones.

collecting QR gesture data from users to inform how a reliable detection algorithm can be designed.

5.2 Related Work

The first force-sensitive touchscreen was developed by Herot et al. [1978] and enabled drawing with a single finger via shear force. Buxton et al. [1985] used continuous force sensing on a touch tablet to control the width of strokes drawn by the finger. Pens for graphic tablets [Mizobuchi et al., 2005; G. Ramos et al., 2004; G. A. Ramos et al., 2007] and force-sensitive mice [Cechanowicz et al., 2007; Shi et al., 2008] have been explored to control desktop applications more efficiently. Force input has also been investigated on resistive handheld touchscreens [Brewster et al., 2009; Wilson et al., 2010], e.g., to toggle small and capital letters during text input. More recently, smartphones and tablets with capacitive touchscreens have been equipped with force-sensing resistors at the bezel [McLachlan et al., 2014; McLachlan et al., 2015; Wilson et al., 2011], sides [Spelmezan et al., 2013b; Wilson et al., 2013], or back [Corsten et al., 2017a] to let the fingers, that hold the device in place, partake in interactions, such as on-screen menu control. Arif et al. [2014]

Using force for human-computer interaction started on stationary input devices and lately has been investigated on handheld devices.

and Arif et al. [2013] exploited users' difference in touch contact time and touch radius when applying various levels of force to the touchscreen to detect two levels of pseudo-force. However, their approach requires the user to always navigate with consistent timings, since longer touch contact time is interpreted as higher force. Various studies investigated the effect of feedback methods [G. Ramos et al., 2004; Stewart et al., 2010; Wilson et al., 2010], transfer functions [Cechanowicz et al., 2007; Shi et al., 2008; Stewart et al., 2010], and discrete force levels [McLachlan et al., 2014; G. Ramos et al., 2004; Stewart et al., 2010; Wilson et al., 2010; Wilson et al., 2011] on users' force input. In summary, users showed good performance with eight to ten discrete force levels on a 3–10 N range using a linear transfer function.

Different confirmation techniques for force input have been designed and tested on various input devices. The most common techniques are *Dwell Time (DT)* and *Quick Release* (QR).

Another influencing factor is the force confirmation technique to confirm force input for a particular value. Most studies used *Dwell Time* (DT) [Brewster et al., 2009; Cechanowicz et al., 2007; Shi et al., 2008; Stewart et al., 2010; Wilson et al., 2011] and *Quick Release* (QR) [Brewster et al., 2009; Cechanowicz et al., 2007; Wilson et al., 2010]. Both force confirmation techniques were first mentioned by G. Ramos et al. [2004], who tested them for force input with a pen on a tablet. While DT has a low and consistent error rate (≈ 3%) [Brewster et al., 2009; G. Ramos et al., 2004; Stewart et al., 2010; Wilson et al., 2011], it slows the interaction, as the user must typically wait for 1 s before her selection is confirmed, resulting in ≈ 2.5 s completion times. Another drawback is that, once force is applied, DT does not allow the user to linger on an item longer than the dwell time. If she needs longer to decide which item to pick, then this leads to unintended confirmation [McLachlan et al., 2014]. In contrast, QR does allow for lingering and is fast (≈ 1 s) [Brewster et al., 2009; G. Ramos et al., 2004; Stewart et al., 2010; Wilson et al., 2011]. However, it has a high and inconsistent error rate (≈ 5–40%) [Brewster et al., 2009; G. Ramos et al., 2004; Stewart et al., 2010; Wilson et al., 2011], and is therefore often ranked lower than DT by users, although they prefer the faster completion time of QR [Brewster et al., 2009; G. Ramos et al., 2004]. Current devices, like iPhone Xs, neither use DT nor QR, but a simple thresholding technique: As soon as the user crosses a particular force value, she reaches the next menu state. This state is maintained as long as the user's force is lower than the next threshold. However, once a threshold is crossed, the user cannot go back, until the threshold is reset, e.g., by tapping on a back button.

Whereas the detection and implementation of DT for discrete force input can be clearly derived from its description, this is less clear for QR. Wilson et al. [2010] identified that "*Designing an accurate Quick Release mechanism is troublesome because it is difficult to identify a common and clear pattern of sensor behaviour from which user intent can be unambiguously retrieved.*". None of the studies using QR explained how they actually detected and implemented it. This might also explain the inconsistency in error rates, given different implementations. Due to this unclarity, we set out to investigate how to design a reliable QR mechanism by studying data from users performing the QR gesture.

So far, there is no clear documentation and understanding of how a reliable QR confirmation technique is implemented. We set out to tackle this unclarity by studying QR gesture data from users.

5.3 Detecting Quick Release

To get a rough estimate when a QR event happens, which is in line with finding out how long it takes the user to lift her finger off the touchscreen, we used the model human processor [S. Card et al., 1986], also known as CMN model: According to this model, a user's action involves three steps, each of which takes a certain time: (1) perception (100 ms), (2) cognition (70 ms), (3) motion (70 ms). In the context of applying the QR gesture to confirm force input for a menu with discrete items, we obtain: (1) the user *perceives* which item is currently selected, (2) she then *decides* to confirm this item, and (3) she then *lifts* the finger off the touchscreen. Hence, a QR event should have happened ≈ 240 ms before the finger was lifted off the screen. However, since the CMN timings are only estimates and may differ by user and context, we conducted a controlled experiment to gather actual data from six participants (aged 24–29, M = 25.83, SD = 2.14, all male, all right-handed) performing the QR gesture. All participants were smartphone users but not experienced with force input.

Finding the value the user wants to confirm means finding the exact time before the user quickly lifted off her finger. We studied this timing in a user study with six participants.

5.3.1 Experimental Design

Users were asked to apply certain force on the touchscreen of a force-sensitive iPhone 6s to select a menu item and then confirm that item by quickly lifting the thumb of the screen. iPhone senses force values between 0 and $\frac{480}{72} \approx 6.67$ in steps of $\frac{1}{72}$ when

We let our select discrete values on a force-sensitive iPhone with their thumb and

asked them to quickly lift off their finger immediately after. Throughout, we continuously logged their force-time data.

force sensitivity is set to "firm". According to Apple's API documentation [Apple Inc., 2019b], a value close to 1.0 is equivalent to an ordinary touch, whereas any higher value indicates intentional force input. The documentation does not state how these values translate to Newtons, but experiments [Nelson, 2015] hint at a ≈ 0–4 N range and a linear transfer function.

The UI showing task instructions to the participants is shown in Figure 5.2.

Figure 5.2 shows the UI of the application that was used for displaying the task to the user and collecting the data. On the left, a vertical menu divided the device force range from 1.0–6.67 equidistantly into discrete segments. Force < 1.0 was not visualized to let ordinary touch input coexist. The segments represented low to high force from bottom to top. When the force applied matched the range of a segment, it was filled with white color. A "+" indicated the segment that the user had to reach via pressing the thumb that should be placed at the location of the circled crosshair. Once the marked segment was reached, it turned green, and the user would immediately lift her thumb off the screen. Then, the next trial was shown.

Independent variables were Menu size, thumb Location, and force Level. Form each participant, we recorded data from 225 trials.

Independent variables were Menu size (5, 10, and 15 segments), thumb Location (Fig. 5.2), and force Level (1.189, 2.418, 3.930, 5.347, and 6.481). Users did not have to reach these values exactly, but stay within the corresponding segment. This way, not all segments were tested, but it ensured that an equal amount of measurements per Menu was obtained—an approach also applied by Corsten et al. [2017a], McLachlan et al. [2014], G. A. Ramos et al. [2007], Wilson et al. [2010], and Wilson et al. [2011]. The choice of Levels guaranteed that for each Menu, the lowest and the highest segment, a segment around the center, a segment between the center and the lowest segment, and a segment between the center and the highest segment were chosen. Each combination from these variables was repeated three times, resulting in 3 × 5 × 5 × 3 = 225 trials per user.

To become familiar with the task, participants performed some test trials first.

Menu was counterbalanced using a Latin Square. Level and Location were combined and randomized. Once all three repetitions of the Level × Location trials were done, the user took a quick break and then continued with the next Menu. To become familiar with the task, each user first performed five test trials when Menu changed, resulting in 3 × 5 = 15 additional trials. The user was sitting in a chair without arm rests and held the phone in his right hand. A session took about 20 minutes.

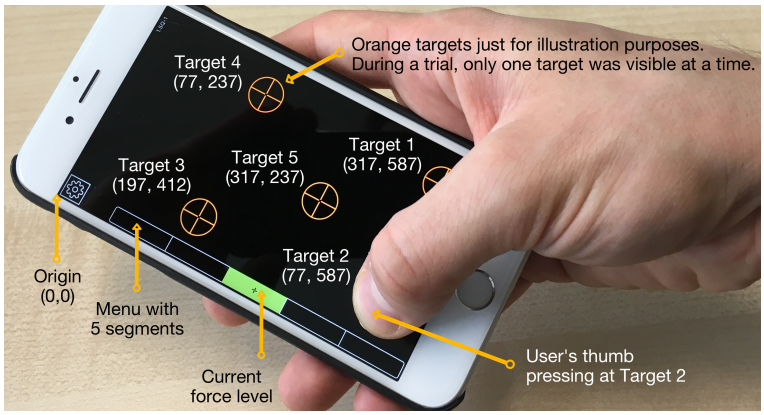


Figure 5.2: UI of the application used to collect data from users performing the Quick Release gesture on a force-sensitive Phone 6s (667 × 375 pt screen).

Dependent variables were *Force* [0–6.67], *Timestamp* [ms], *Frame* [i, $i \in \mathbb{N}_{\geq 1}$], and *Success* [0,1]. *Frame* denoted the i^{th} frame from the touch digitizer stream, which is sampled every ≈ 16 ms on iPhone. *Success* denoted whether the *Force* of a *Frame* was within the force range represented by the marked segment (1) or not (0).

Dependent variables were continuously logged at 60 Hz, i.e., every 16 ms.

5.3.2 Results

Since a QR event happens right before the end of a trial, we reversed our data, such that the *Timestamp* from the last *Frame* received from the digitizer (`touchesEnded` call in iOS) was set to 0 ms. This way, we could align all trials to the same final *Timestamp*, independent of how long the user needed to complete a trial. For convenience, we refer to discrete reversed *Frame*s in our analysis. Yet, multiplying *Frame* by ≈ 16 ms yields the equivalent continuous *Timestamp*.

We reversed the sequence of the logged force-time *Frame*s to obtain normalized data.

Figure 5.3 shows a typical plot of *Force* vs. reversed *Frame* for one user for Menu 5. The green color indicates that the user's *Force* (y-axis) matched the requested segment at the time the *Frame* (x-axis) was captured. The data shows that most matches are found around reversed *Frame* 16. This was generally independent from Level and Menu: Figure 5.4 shows the cumulated count of *Suc-*

Looking at the force-time *Frame*s in reversed order, about 256 ms before the thumb was completely lifted off the

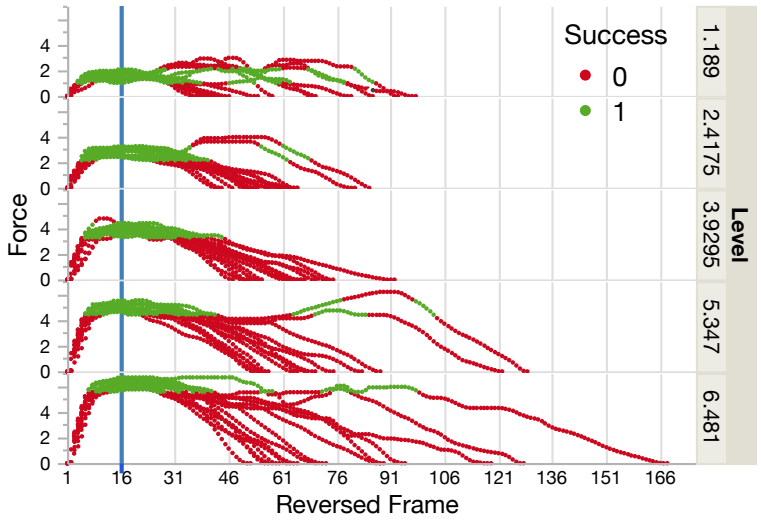


Figure 5.3: A typical plot showing reversed *Frame* vs. *Force* data from the Quick Release gesture performed by one user for MENU 5. Around reversed *Frame* 16, the user's force always matched the requested LEVEL. Green points from higher reversed *Frame*s indicate that the user reached the correct LEVEL, but did not release yet, e.g., because she was jittering

touchscreen, the users' force matched the requested LEVEL segment. This finding was independent from MENU and LEVEL.

cess per reversed *Frame*. For each MENU there is a clear peak, and all three peaks almost overlap. If we use a *Frame*-to-*Force* lookup for the *Frame* at each peak and map the *Force* to the corresponding segment, we obtain promising success rates of 96% for MENU 5, 98% for MENU 10, and 95% for MENU 15. However, using just a single reversed *Frame* for the lookup might be problematic, especially when the user was jittering.

We implemented an algorithm that identifies a sequence of adjacent reversed *Frames* for which overall *Success* is highest taking all trial data into consideration.

Therefore, we were interested in identifying a sequence of *Frame*s that lead to the highest *Success* rate for each MENU SIZE. We applied the following optimization process: Assume that for one trial, $n \in \mathbb{N}_{>0}$ *Frames* were recorded. Then let F_i, $i \leq n$ be the i^{th} reversed *Frame* and let $w \in W = \{0...min(i-1, n-i)\}$ denote a *width*. Then $F_i(W) = \{F_i(w)\}$ is a set of sequences $F_{i-w}, \ldots, F_{i-1}, F_i, F_{i+1}, \ldots, F_{i+w}$ of $2w+1$ reversed *Frames* with F_i at the center. For each F_j in $F_i(w)$ $(1 \leq j \leq i+w)$, we calculated the menu segment $M(F_j)$ for the *Force* measured at that *Frame*. $Success(M(F_i(w))) \in \{0, 1\}$ denoted whether the segment in $\{M(F_i(w))\}$ that occurred most frequently matched the

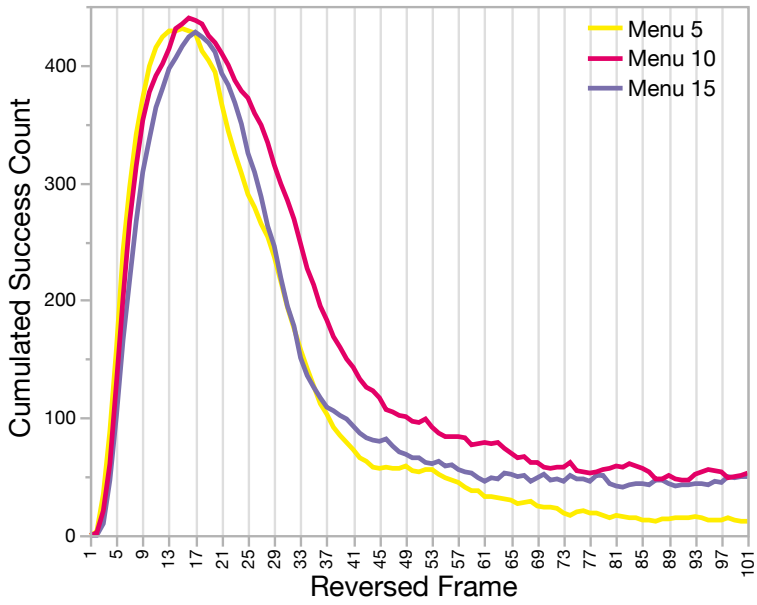


Figure 5.4: The cumulated count of frames from trials that successfully matched the requested LEVEL for all three MENU sizes. The Quick Release timing is likely to be independent from LEVEL and MENU since all three plots are almost identical.

requested segment for a trial. Figure 5.5 shows the success rates for MENU for $5 \leq i \leq 30$ and $0 \leq w \leq 30$. Although wider ranges for i and w could have been chosen, the peaks in Figure 5.4 indicated that the optimum should be located within those ranges. As can be seen, there are multiple but few combinations of parameters i and w at the peaks of the curves. Although each MENU has a different F_i that leads to the maximum success rate (MENU 5: 96% for i =15, MENU 10: 98% for i =16, MENU 15: 97% for i =17) they share an optimal width of $w = 3$. Converting these sequences of $F_i(3)$ into time ranges, we obtain 192–288 ms for MENU 5, 208–304 ms for MENU 10, and 224–320 ms for MENU 15. As assumed, the 240 ms calculated from the CMN model are always contained within these ranges. As a next step, we wanted to see how users' performance for force confirmation with QR using these optimal parameters compares to confirmation with DT.

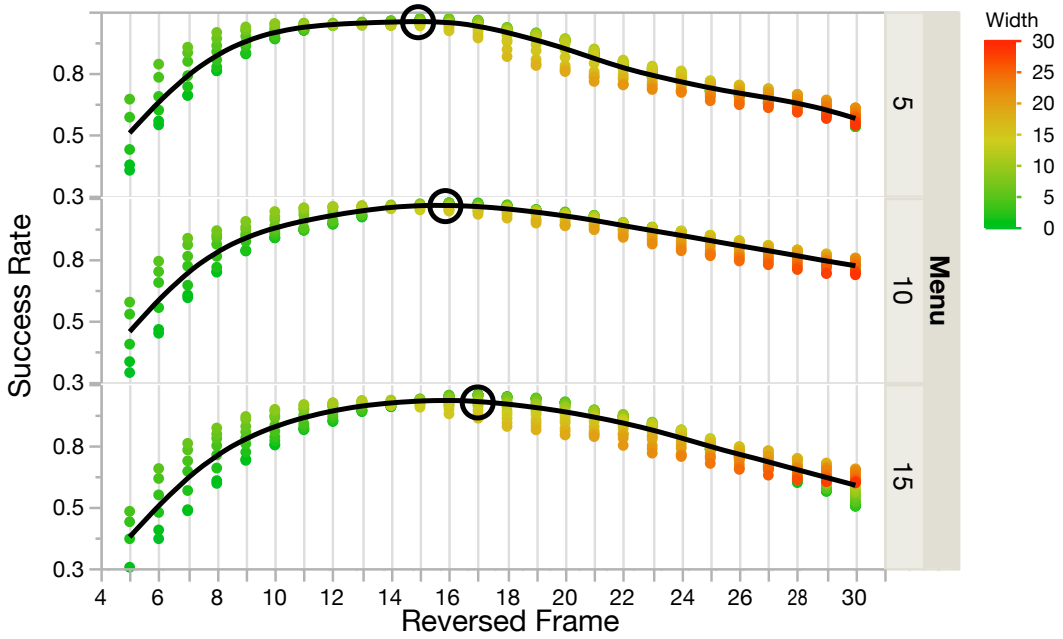


Figure 5.5: Plots of varying *Success* rates for different combinations of reversed *Frame* and *width* for the Quick Release gesture data. Circles indicate reversed *Frame*s for MENU that led to optimal *Success* with a *width* of $w = 3$.

5.4 Verification Experiment

To test our QR algorithm against classic DT, we conducted a verification experiment with fresh participants.

We extended our first experiment to include QR as force confirmation technique using our previously determined parameters. The segment that occurred most frequently in $M(F_i(w))$ was feedbacked to the user as soon as she lifted her thumb. Half a second later, the next trial was shown. For baseline comparison, we also added DT: If the user maintained force for a segment for at least one second, it was confirmed through orange color. Lifting the thumb initiated the next trial. Twelve users (aged 23–53, $M = 32.33$, $SD = 8.17$, four female, all right-handed) participated. All participants were smartphone users but not experienced with force input, and none of them had participated in our previous experiment.

The study design was similar to the first experiment, but for testing against

Independent variables from the first experiment were extended by TECHNIQUE, which denoted the two force confirmation techniques. Thumb LOCATIONS were slightly changed to test robustness against thumb placement. We did not change LEVEL

since Menu 5 already included all five possible segments, and for Menu 10 and 15 a change of Level would only have resulted in segments adjacent to the original ones. Counterbalancing and randomization were inherited; half of the users started with DT. Each participant performed 2 Technique × 3 Menu × 5 Level × 5 Location × 3 repetitions = 450 trials. Before the next Menu was tested, the user performed five test trials, resulting in 2 × 3 × 5 = 30 additional trials.

robustness of our algorithm, we slightly changed the users' placement of the thumb Location for each trial.

Dependent variables were *Success* ∈ [0, 1] and *Time* [ms]. *Success* denoted whether the requested and the predicted segment matched (1) or not (0), and *Time* was measured until a trial was completed, i.e., until the dwell time expired or, for QR, when the thumb was no longer in contact with the touch digitizer. At the end, users were asked to vote for their preferred force confirmation technique, or whether they had no preference at all.

Dependent variables were *Success*, i.e., whether the requested Level was confirmed by the QR and DT algorithms, and the task completion *Time*.

5.4.1 Results

Since we interested in a direct comparison between DT and QR, we always directly contrast both Techniques for each variable. *Time* was log-transformed for repeated-measures ANOVAs. For *Success*, we conducted McNemar and Cochran's Q tests, since the data was dichotomous.

In our analysis, we directly contrasted DT vs. QR performance.

Technique had a significant effect on ***Time*** ($F_{1,5377}$ = 348.60, p < .0001): Using DT, users needed 2,341 ms (95% CI: ±51 ms) to complete a trial on average, which was twice as slow compared to 1,125 ms (95% CI: ±26 ms) for QR.

Using QR, participants were twice as fast compared to using DT.

Menu had a significant effect on *Time* for both DT ($F_{2,2686}$ = 159.02, p < .0001) and QR ($F_{4,2686}$ = 179.80, p < .0001). Tukey HSD post hoc pairwise comparisons for each Technique were all significant, i.e., *Time* increased with Menu for both, DT and QR. Pairwise t-tests of *Time* between the same Menus across both Techniques were all three significant ($F_{4,1787}$ ≥ 1121.34, adjusted p < .001, each), i.e., for each Menu, confirmation with QR was significantly faster (Fig. 5.6).

Time increased with Menu size. For each Menu size, participants were always faster with QR.

Level had a significant effect on *Time* for both DT ($F_{4,2684}$ = 160.78, p < .0001) and QR ($F_{4,2684}$ = 158.84, p < .0001).

Participants were fastest for the lowest

Level		1.189	2.4175	3.9295	5.347	6.481
Mean	DT	2,228	2,189	2,623	3,071	1,591
	QR	956	973	1,288	1,540	869
95% CI ±	DT	107	105	108	136	48
	QR	49	42	64	74	37
Target		**1**	**2**	**3**	**4**	**5**
Mean	DT	2,663	2,248	2,325	2,189	2,279
	QR	1,138	1,088	1,130	1,139	1,132
95% CI ±	DT	140	97	113	93	109
	QR	62	54	59	61	57

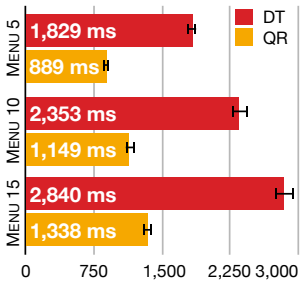


Figure 5.6: Dwell Time vs. Quick Release results for *Time* in ms. Error bars denote 95% CI.

and the highest Level. The lowest Level is the first item, thus fast to acquire. The last Level is also quickly acquired by applying maximum detectable force, i.e., the participant does not need to worry about overshooting.

For DT, Tukey HSD post hoc pairwise comparisons were not significant between the two lowest Levels, but between all other pairs ($p < .0001$, each). For QR, Tukey HSD post hoc pairwise comparisons were also not significant between the two lowest Levels and between the lowest and the highest Level—for these, users were fastest—, but between all other pairs ($p < .0001$, each) (Fig. 5.6). Being faster at lower Levels is plausible since applying more force takes more time, and for the highest Level, users could quickly apply as many force as they could to always land on the segment for the highest force level. Pairwise t-tests for *Time* between the same Levels across both Techniques were all significant ($F_{1,1067} \geq 626.80$, adjusted $p < 0.001$, each). For any Level, force confirmation with QR was always significantly faster than with DT.

For both, QR and DT, participants were slowest when applying force from the lower left corner of the touchscreen.

Location had a significant effect on *Time* for DT ($F_{4,2684} = 9.03$, $p < .0001$): Tukey HSD pairwise comparisons showed that confirming force input with the thumb at the lower left corner of the screen was significantly slower than any other Location ($p < .001$, each). For QR, however, Location had no effect on *Time* ($F_{4,2684} = .97$, ns.). Pairwise t-tests for *Time* between the same Locations across both Techniques were all significant ($F_{1,1067} \geq 656.03$, adjusted $p < 0.001$, each). For each Location, confirmation with QR was significantly faster (Fig. 5.6).

Technique had no significant effect on *Success*.

Technique had no significant effect on ***Success*** ($\chi^2(1) = .35$, ns.), i.e., we could not prove that any Technique performed better than the other as regards *Success*.

Level		1.189	2.4175	3.9295	5.347	6.481
Mean	DT	97.41	97.59	96.85	94.63	99.81
	QR	96.48	98.70	96.11	96.67	99.81
95% CI ±	DT	1.35	1.29	1.48	1.91	.36
	QR	1.56	.95	1.63	1.52	.35
Target		**1**	**2**	**3**	**4**	**5**
Mean	DT	98.33	97.96	96.67	96.49	96.85
	QR	97.04	97.78	97.04	97.78	98.15
95% CI ±	DT	1.08	1.19	1.52	1.57	1.48
	QR	1.44	1.25	1.44	1.25	1.14

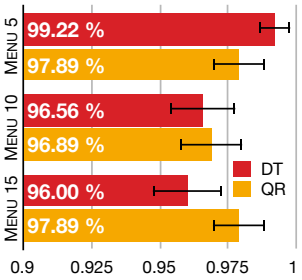


Figure 5.7: Dwell Time vs. Quick Release results for *Success* in %. Error bars denote 95% CI.

Menu had a significant effect on *Success* for DT ($Q(2) = 20.60$, $p < .0001$). Post hoc pairwise comparisons revealed that *Success* for Menu 5 (99.2%) was significantly higher compared to Menus 10 (96.6%) and 15 (96.0%) ($p < .001$, each). For QR, however, Menu had no effect on *Success* ($Q(2) = 2.49$, ns.)—users performed equally well independent from the Menu size. Pairwise comparisons of *Success* between the same Menus across both Techniques were significant for Menu 5 ($\chi^2(1) = 4.65$, $p < .05$) and for Menu 15 ($\chi^2(1) = 4.66$, $p < .05$): Whereas *Success* for Menu 5 was significantly higher for DT (99.2%) than for QR (97.9%), *Success* for QR was significantly higher for Menu 15 (97.9%) compared to DT (96.0%). Yet, these differences were smaller than 2% (Fig. 5.7).

Only for QR, Menu had no effect on performance, i.e., with an average *Success* rate of 97%, our algorithm performed equally well for all Menu sizes.

Level had a significant effect on *Success* for DT ($Q(4) = 28.23$, $p < .0001$). Post hoc pairwise comparisons revealed that *Success* for Level 5.347 was significantly lower compared to all other Levels except for Level 3.925, for which, however, *Success* was significantly lower compared to Level 6.481 ($p < .05$, each). Level also had a significant effect on *Success* for QR ($Q(4) = 24.38$, $p < .0001$). Pairwise McNemar tests showed that *Success* for Level 6.481 was significantly higher compared to all other Levels except for Level 2.4175 ($p < .001$, each). Pairwise comparisons of *Success* between the same Levels across both Techniques were all not significant (Fig. 5.7).

Level had a significant effect on *Success* for both, DT and QR.

Location neither had a significant effect on *Success* for DT ($Q(4) = 5.81$, ns.) nor for QR ($Q(4) = 2.25$, ns.). Likewise, pairwise comparisons of *Success* between the same Locations across both Techniques were also all not significant (Fig. 5.7).

The thumb Location has no effect on *Success* for both, DT and QR.

Participants preferred using QR over DT.

Users' ***preference*** for TECHNIQUE was significant ($Q(2) = 9.50$, $p < .001$): Nine users voted significantly higher for QR compared to two votes for DT and one for 'no preference' ($p < .05$, each).

5.4.2 Discussion

Overall, QR is the more advantageous technique, since it is faster and more preferred by participants, while both, QR and DT share the same 97% *Success* rate.

Not surprisingly, users were faster with QR than with DT; results for *Time* were in line to findings from related work. Interestingly, using DT took 1,216 ms longer—216 ms more than the added DT. This could be explained by a strategy from users, who increased or decreased force very slowly when they jittered around a targeted segment. Also note that *Time* for DT stopped right after the timer expired, but in practice, the user would still need to lift her finger to continue interaction. This would add another 240 ms from the CMN model, which is, on the contrary, already included in *Time* for QR. Although LOCATION did not influence *Success*, users were slower with DT when their thumb was placed at the lower right corner of the touchscreen. Maintaining force at that location is difficult, as the device is imbalanced and the thumb is folded. The QR gesture, however, did not involve maintaining force, which might explain why users were equally fast at any LOCATION. *Success* for DT was in line with findings from related work. Yet, we could not find an overall difference between DT and our implementation of QR. What is more, is that unlike DT, QR was not influenced by MENU, which makes it more flexible as regards the choice of menu size. Combined with users' faster interaction time and higher preference for QR, we can summarize it as the more advantageous force confirmation technique compared to DT. We envision QR to allow for effective control of context menus, e.g., to quickly access shortcuts in smartphone applications.

5.5 Limitations and Future Work

Since the confirmed item is calculated after complete thumb lift-off, the menu cursor will rapidly

We collected all data on iPhone 6s. However, we expect similar results for other devices, since we saw that the QR gesture is in line with predictions from the device-independent CMN model [S. Card et al., 1986]. We also plan to optimize feedback visualization for QR: As of now, when the user lifts her thumb, the

menu cursor will rapidly drop and then jump back to the confirmed segment. This is since we cannot start calculating which segment the user wanted to confirm before she completely lifted her thumb. A solution could be pausing visualization updates when a rapid decrease in force ($\delta_{Force} \leq$ -0.3 units on the 0–6.67 scale) for QR is detected, but this would probably be dependent from the sensor range and menu items.

drop and then jump back to the confirmed segment. We plan to optimize this feedback artifact by pausing feedback updates.

5.6 Conclusion

We set out to challenge the low reliability of Quick Release (QR), a common technique to confirm force input by quickly lifting the finger. We contrasted it with Dwell Time (DT), a technique that requires the user to maintain pressure for 1 s to confirm the input, and that is reported as more reliable than QR. While detecting and implementing DT is clear and straightforward, literature does not describe how to do so for QR. Inspired by the CMN model [S. Card et al., 1986], we hypothesized that the force the user intends to confirm can be retrieved by looking ≈ 240 ms back in time once the finger has been lifted. To confirm our hypothesis, we collected data from users who controlled menu items on a force-sensitive smartphone by pressing with the thumb. Based on this data set, we implemented an algorithm to detect QR: Use a force-to-menu item lookup between ≈ 200–300 ms before the finger is lift off the screen and pick the item that occurred most frequently within that time frame. In a verification study, we tested this implementation against DT: With a 97.6% success rate, QR was as reliable as DT, that had a 97.2% success rate. Combined with users' faster performance and higher preference for QR, we recommend it over DT as confirmation technique for force input on modern smartphones.

We presented an algorithm for reliably detecting when the user quickly lifts her thumb off a force-sensitive touchscreen to confirm her force input. To determine and verify this QR algorithm, we collected gesture data from participants. In contrast to DT, our QR implementation achieved the same accuracy, but was faster and more preferred by our participants.

With this improved version of Quick Release, we now have an efficient and reliable confirmation technique for one-handed force input on handheld devices. Next, we will present an interaction technique that embeds our Quick Release implementation and solves common issue for one-handed smartphone use: reachability. With growing screen sizes due to increasing device sizes

Next, we will look at an interaction technique that uses our Quick Release technique to solve a recurring issue for

one-handed device use: limited reachability of the user's thumb.

and shrinking screen bezels, users cannot reach everywhere on the touchscreen without stretching their thumb and re-grasping the device. In the next chapter, we present ForceRay, an interaction technique, that tackles this reachability issue with force input.

6

Extending Thumb Reach on Handheld Devices via Force Touch Input

→ Summary

Smartphones are used predominantly one-handed, using the thumb for input. Many smartphones, however, have grown beyond 5". Users cannot tap everywhere on these screens without destabilizing their grip. We designed *ForceRay* (FR), an interaction technique that lets users aim at an out-of-reach target by applying a force touch at a comfortable thumb location, casting a virtual ray towards the target. Varying pressure moves a cursor along the ray. When reaching the target, quickly lifting the thumb selects it. In a first study, FR was 195 ms slower and had a 3% higher selection error than the best existing technique, BezelCursor (BC), but FR caused significantly less device movement than all other techniques, letting users maintain a steady grip and removing their concerns about device drops. A second study showed that an hour of training speeds up both BC and FR, and that both are equally fast for targets at the screen border.

Publications: The work presented in this chapter has been done in collaboration with Marcel Lahaye, Simon Voelker and Jan Borchers. The author of this thesis developed the research idea and relevant research questions, including the motivation of the work. Furthermore, he has designed all experiments, partially implemented them, and analyzed all data from the conducted user studies. Most of this work has been published as paper in the Proceedings of ACM CHI 2019 [Corsten et al., 2019]. The author of this thesis is the main author of the paper. Most sections in this chapter are taken from the paper publication.

6.1 Motivation

Since force input increases the expressiveness for a single touch at a fixed location, we wondered whether it can be useful to solve reachability issues on modern smartphones.

Knowing now how input on force-sensitive handheld touchscreens can be reliably and efficiently confirmed, we can make use of it as an additional dimension of touch input on handheld devices with force-sensitive touchcreens. Since force input has the benefit of providing one-dimensional input without moving the finger, we wondered whether it can be used to solve a recurring problem of users not being able to reach all touch targets on a handheld device. This is typically the case for one-handed use since then users can only use their thumb for touch input.

Using a single hand for touch input, users cannot reach the entire touchscreen with their thumb without changing their grip. With growing device sizes, this reachability issue becomes more prevalent.

Over the years, smartphone touchscreens have been growing in size [Fingas, 2012], from 3.5" in Apple's original iPhone from 2007 to 6.5" in their current iPhone Xs Max, for example. While larger screens can show more content at a time, they are a mixed blessing for touch input: Users tend to interact with their smartphones using a single hand, whether due to user preference [Boring et al., 2012; Hirotaka, 2003] or because the other hand is holding a coffee cup, carrying a bag, or holding on during a train ride. This leaves the thumb as the only finger to interact with the touchscreen [Karlson et al., 2008b]. This way, however, the user cannot comfortably reach all parts of the screen unless she re-grasps the device, which is inconvenient and takes time. Most critically, re-grasping destabilizes the device grip and causes increased device motion. This makes users feel insecure in holding their device [Eardley et al., 2017; Eardley et al., 2018b] and can lead to accidental drops, breaking the screen or other components. This out-of-reach area grows with screen size.

Common solutions to the reachability issue require frequent mode switching or reduce screen content.

Industry and HCI research have proposed several ways to mitigate this reachability issue (cf. Section 6.2), but these approaches require explicit mode switching, allow using only a small part of the screen for interaction, or still cause significant device motion while selecting targets.

We designed an interaction technique, called *ForceRay* (FR) that solve all these issues. By applying a force touch, a ray is

We designed *ForceRay* (FR), a reachability technique that addresses these issues. It uses the force sensing touchscreen found in recent smartphones: When the user applies a force touch with her thumb on the touchscreen, a ray appears that points from the lower screen corner under her palm through her thumb's touch position to the opposite edge of the screen (Fig. 6.1). If necessary,

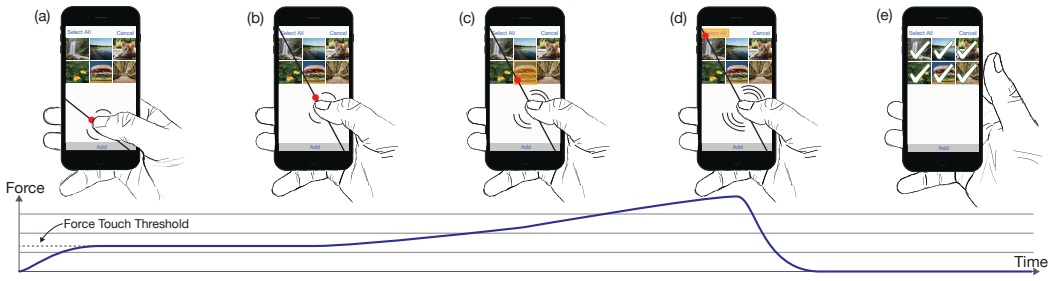


Figure 6.1: The user wants to tap the button in the top left corner of her smartphone screen to select all images at once. Reaching this button, however, is hardly possible without changing the device grip, which leads to increased device motion and thus is likely to result in a device drop. The *ForceRay* interaction technique addresses this reachability issue without destabilizing the device grip: (a) When the user applies a force touch on the screen, a ray is displayed that extends the user's thumb reach to the opposing screen edge. (b) The user drags her thumb to reposition the ray until it crosses the desired target. (c) To reach the button, the user increases her force to move the red cursor along the ray. The cursor automatically highlights the first selectable target, an image, on the ray. The more force is applied, the farther the cursor moves along the ray, always highlighting the underlying target. (d) At maximum force, the 'Select All' button is reached. (e) Lifting the thumb triggers the button.

she can roll her thumb left or right to fine-tune the direction of the ray so that it crosses the intended target. Along this virtual thumb extension, she controls a cursor: The more force she applies, the further the cursor moves away from the thumb. If the cursor exits a target, the next target along the ray is highlighted automatically. To select it, the user quickly lifts her thumb off the screen. To cancel instead, she reduces her force below the force touch activation threshold before lifting the finger.

casted towards a subset of targets. To select a distant target the user applies force to move a cursor on that ray towards the desired target. QR confirms the selection.

FR is a "quasi-mode" [Raskin, 2000]: It is active only while the user is consciously maintaining her force touch, thus avoids confusing and time-consuming mode switching through other explicit input gestures. Furthermore, it benefits natural, ergonomic thumb movement that enables the user to maintain a stable device grip. FR's design scales to different screen sizes and form factors, and makes it especially fast to select targets at screen borders by simply applying maximum force.

FR is only active as long as contact with the touchscreen is maintained. It also supports the ergonomic movement of the thumb.

Thus, the key contribution of this chapter is the FR interaction technique that extends thumb reach via force input to enable selection of out-of-reach targets with a steady device grip. We first

We evaluated FR against other reachability

techniques in two user studies.

review related work before describing the design and implementation of FR. To validate FR, we present two user studies: Study 1 compared FR to standard direct touch input and three existing reachability techniques: the *One-Handed Mode*, found in Android devices, that downscales and moves the screen towards the user's thumb, *MagStick* [Roudaut et al., 2008], and *BezelCursor* (BC) [W. H. A. Li et al., 2013]. Among all, FR significantly caused the least device motion. Study 2 showed that an hour of training sped up FR selection time and that users selected far targets at the screen border as fast as the fastest candidate from Study 1, BC, with 96% selection accuracy. We close with recommendations to address reachability issues for one-handed touchscreen use.

6.2 Related Work

FR addresses reachability and force input techniques, both, on handheld devices. We discuss the related work successively.

6.2.1 Reachability Techniques

Bergstrom-Lehtovirta et al. [2014] modeled up to where the thumb can comfortably reach on handheld touchscreens.

Probably the most straightforward approach is to constrain the UI layout to just the comfort region of the thumb. With this in mind, Bergstrom-Lehtovirta et al. [2014] built a model that predicts where the user's thumb can reach. However, this limits the space for interactive elements to a constant area in one screen corner, ignoring the extra available space beyond.

Chang et al. [2015] constructed a design space for reachability techniques for handheld device interaction.

To survey solutions that address reachability for UIs laid out on the entire screen, Chang et al. [2015] constructed a design space that classifies interactions by their *trigger* and *targeting* mechanisms. Trigger mechanisms look at how the technique is activated, and targeting mechanisms address how a target is selected. The latter distinguish between techniques that apply a *screen transform*, provide a *proxy region*, and use a *cursor* to select a target. We follow this useful taxonomy.

Screen Transform Techniques displace or

Screen Transform Techniques. Smartphone manufacturers embed such techniques in their operating systems. In iOS,

shrink the UI on the handheld touchscreen to move it closer to the user's thumb.

double-tapping the Home button or swiping down across the bottom screen edge slides the screen half down, but this leaves targets on the far side opposite of the thumb unreachable. Samsung's *One-Handed Mode* shrinks the entire screen to be close to the thumb when triple-tapping the Home button or sliding from the corner. TiltReduction [Chang et al., 2015] shrinks the screen likewise when tilting the device. Sliding Screen [Kim et al., 2012] moves the screen diagonally towards the thumb, and is either triggered by swiping or by generating a large touch footprint. TiltSlide [Chang et al., 2015] works similarly, but is activated by tilting the device. MovingScreen [Tsai et al., 2016] also moves the screen to the thumb, depending on how far the user swipes from the screen edge. Le et al. [2016] trigger the same effect by sliding the index finger across a touchpad at the back of the device (BoD). Löchtefeld et al. [2015] detect which hand unlocked the device to shift the UI towards that hand. Eardley et al. [2017] and Eardley et al. [2018a] present several adaptive UI concepts, e.g., a keyboard that shifts to the user's thumb when the device is tilted sideways.

However, these techniques hide context information or make targets too small to read and hit.

All these techniques, however, either omit parts of the UI and thus hide context information, or shrink targets, making targets more difficult to hit, or need hardware modifications, or they use tilt, which is prone to overshooting and makes reading the screen difficult at certain angles [Spelmezan et al., 2013b].

Proxy Region Techniques provide a small proxy through which the user can interact with the entire screen.

Proxy Region Techniques. TapTap [Roudaut et al., 2008] magnifies a part of the screen around the thumb's touch location in a pop-up view. ThumbSpace [Karlson et al., 2007; Karlson et al., 2008a] uses a similar concept but here, the view represents the entire screen, making targets very small and difficult to hit. Both techniques do not scale well to large form factors and their proxy views occlude a part of the screen. Hasan et al. [2016] proposed the mid-air space between thumb and touchscreen as proxy region. Löchtefeld et al. [2013] used a BoD touchpad to reach upper targets with the index finger from behind. Such proxy region extends the thumb's reach by 15% [Yoo et al., 2015]. However, these techniques require hardware modifications.

Cursor Techniques let the user select out-of-reach targets via a cursor.

Cursor Techniques. TiltCursor [Chang et al., 2015] lets users drag an accelerated cursor with the thumb and is triggered by tilting. BezelCursor [W. H. A. Li et al., 2013] is similar but activated by swiping from the bezel, which overwrites system-wide

triggers, e.g., for opening the phone's control panel. ExtendedThumb [Lai et al., 2015] is triggered by a double tap and—similar to BezelCursor [W. H. A. Li et al., 2013]—extends the thumb with a cursor whose offset increases with dragging speed. Extendible Cursor [Kim et al., 2012] is similar but steers the cursor opposite to where the user drags her thumb. MagStick [Roudaut et al., 2008] also uses this opposite dragging mechanism to avoid occluding the target with the thumb and is triggered when pressing on the screen. However, the swiping that these techniques require, can lead to grip instability when the thumb is moved beyond its comfort region. 2D-Dragger [Su et al., 2016] solves this problem by stepping through UI elements by tiny swipe gestures, which, however, is tedious and time-consuming. CornerSpace and BezelSpace [Yu et al., 2013] also require only little thumb movement to reach targets at screen corners and edges and are triggered by a bezel swipe: BezelSpace lets the user control a cursor like an extended fingertip though a proxy region. CornerSpace initially places a remote cursor at corners to access them quickly and allows for selecting nearby targets through dragging with automatic snapping. However, these techniques make selection of other targets less accurate. Dual-Surface Input [Yang et al., 2009] uses a BoD touchpad with an absolute mapping to select screen targets by tapping with the index finger at the BoD. Yet, like other BoD solutions, hardware modifications are needed.

6.2.2 Force Input Techniques

While there exit various interaction techniques based on force input, none of them has used force as opportunity to solve the reachability issue on handheld devices.

Force adds a third dimension to touch. Davidson et al. [2008] and Qiu et al. [2016] use force to move objects along the z-dimension on the touchscreen. Apple's 3D Touch [Apple Inc., 2018] uses force as a modifier to pop up context menus. Boring et al. [2012] use the thumb's contact size as simulated force to toggle panning and zooming on a smartphone. Goel et al. [2012] discriminate three levels of simulated force from gyroscope readings when the user is pressing against the touchscreen while the smartphone vibration motor is pulsed. Brewster et al. [2009] use a force modifier to conveniently type uppercase letters, and ForceBoard [Zhong et al., 2018] maps force intensity to character selection to type on a smartphone keyboard while looking at a distant screen. ForceEdge [Antoine et al., 2017] maps

force to the velocity of scrolling though long lists. For smaller lists and menus, force is typically mapped directly to selectable values. This way, users can control six to ten items on a 3–10 N pressure range with visual feedback in a stationary context [Cechanowicz et al., 2007; McLachlan et al., 2014; Mizobuchi et al., 2005; Stewart et al., 2010; Wilson et al., 2010]. Other examples control such menus from the bezel [McLachlan et al., 2014; Spelmezan et al., 2013a; Spelmezan et al., 2013b; Wilson et al., 2013] or BoD [Corsten et al., 2017a] with the fingers grasping the device. Hence, force input keeps hand and fingers in place to enable a stable device grip.

None of these solutions exploited force input benefits to solve reachability issued for one-handed smartphone use.

6.3 The ForceRay Interaction Technique

FR consists of a few interaction steps: (1) activation via force touch, (2) placement of the virtual ray on a target subset, (3) cursor control via force until the desired target is hit, and (4) confirmation of the selection by QR.

ForceRay (FR) follows a simple sequence of small interaction steps: When the user places her thumb on the touchscreen without applying significant force, FR is inactive. Once the user starts applying force and crosses a *resting threshold* (this is known as a *force touch*), FR is activated as long as the user maintains force above that threshold. Upon activation, a ray is displayed that virtually extends the user's thumb to the opposing screen edge (Fig. 6.1a). The ray's placement is determined by the touch location and a reference point that represents the thumb's carpometacarpal (CMC) joint location. This is the joint around which the user rotates her thumb to steer the ray, and is approximately located in the lower right (left) screen corner for right (left) handed users.

Targets at edges and corners can be hit by the cursor by applying as much force as possible once the target is crossed by the ray.

Usually, the user will initiate the ray at a point that makes the ray already go through the intended target. If she misses, she can simply move her thumb to the left or right to reposition the ray until it intersects with the target (Fig. 6.1b–c). With the ray comes a cursor that the user controls via force input to highlight any target that intersects with the ray. The more force is applied, the farther the cursor moves upwards on the ray towards the opposite screen edge (Fig. 6.1c–d); reducing the force makes the cursor move downwards again. To make this mapping consistent, i.e., have the same change in force always result in the same

cursor travel distance, FR maps the available force range between force threshold and maximum sensible force to the length of the touchscreen diagonal, which is the longest possible ray length. A mapping that always maps the same force to the same value, known as *Positional Control*, is often applied to force input (e.g., Wilson et al. [2010]) and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

FR's cursor position can be expressed in polar coordinates.

Mathematically, FR's cursor position is a polar coordinate (r, Φ): r is the distance to the CMC reference point, and Φ is the angle between the lower screen edge and the ray.

The user does not need to place the cursor onto the desired target. Once the cursor exits a target, the next target on the ray is automatically selected. To confirm, the user performs the QR gesture, i.e., she quickly lifts her thumb off the touchscreen.

To make target highlighting more efficient, FR does not require the user to move the cursor precisely onto the target: Since the targets the ray crosses follow a defined sequence in which the cursor will reach them, the next target can already be highlighted before actually entering it (Fig. 6.1c–d). In addition, the segment to the first target on the ray is associated with the first target, and the ray segment beyond the last target is associated with the last target. Such virtual cursor enlargement is similar to the idea behind techniques like Bubble Cursor [Grossman et al., 2005] or DynaSpot [Chapuis et al., 2009]. Cursor enlargement not only speeds up the highlighting process, but also avoids invalid selections. To finally select the highlighted target, the user quickly lifts her thumb (Fig. 6.1e). This selection mechanism, also known as *Quick Release,* is a common technique to confirm a selection with force input (e.g., Brewster et al. [2009], Corsten et al. [2017b], G. Ramos et al. [2004], and Wilson et al. [2010]). If, however, the user wants to cancel without selecting a target, she reduces force until it is below the force touch threshold and then lifts her thumb.

FR has three key benefits:

1. **Scalability.** It scales to arbitrary handheld screen sizes since all targets can be reached from within the functional area of the thumb, given that the force sensor is strong enough such that the user can select each target on the ray.

2. **Efficiency.** Independent from the screen size, FR makes selection of targets located at corners and edges efficient: applying maximum possible force immediately highlights

the most distant target crossed by the ray. The iOS back button located in the upper left screen corner is an apt example illustrating this benefit: it is notoriously hard to reach with one-handed input otherwise, but becomes very quick and easy to select with FR.

3. **Visibility.** With FR, the user does not occlude content of interest with her thumb, since only the thumb's functional area is partially occluded.

6.4 Study 1: Reachability Techniques

We wanted to understand how users perform with FR against classic *Direct Touch (DT)* input and other a few other existing reachability techniques: *One-Handed Mode (OM)*, *MagStick (MS)*, and *BezeCursor (BC)*.

To understand how FR compares to the state of the art, we conducted a user study with 15 participants (21–33 years, M = 25.73, SD = 3.31; two female; all right-handed; thumb length: M = 75.87 mm, SD = 7.12 mm). They were all smartphone users (screen size: M = 5.36", SD = .46"). We compared FR to One-Handed Mode (OM, similar to Samsung's), MagStick (MS) [Roudaut et al., 2008], and BezelCursor (BC) [W. H. A. Li et al., 2013] to cover a good variety of techniques: OM is a common commercial solution, MS is one of the first handheld reachability techniques and often compared to by other papers, and BC is well scalable. We added Direct Touch (DT) input as baseline. We asked users to select targets with their thumb on a handheld touchscreen using each of these techniques while holding the device in their right hand in portrait orientation.

6.4.1 Apparatus, Techniques, and Task

For the study task, we used a force-sensitive iPhone 6s as handheld device with a touchscreen size of 118 × 66 mm.

We used an iPhone 6s Plus to present the task to our users and to capture data. For our implementation of FR, we used the force readings provided by the force-sensitive touchscreen. According to Apple's documentation [Apple Inc., 2019b], the force sensor API delivers unitless force values between 0 and $\frac{480}{72} \approx 6.67$ in steps of $\frac{1}{72}$, with force sensitivity set to "firm". Values around 1.0 should be interpreted as an ordinary touch; higher values as intentional force input. Although Apple does not state how these values translate to Newtons, experiments [Nelson, 2015] suggest a 4 N range and a linear transfer function. Although FR combines dragging while exerting force, friction is low due to the small 4 N

force range and the smoothness of the touchscreen glass surface. The iPhone screen measured 736 × 414 pt. For iPhone 6s/7/8 Plus, 1 pt is about .16 mm.

Techniques

OM shrinks the entire UI on the screen to 2/3 and moves the UI towards the user's thumb.

- **One-Handed Mode (OM)** mimics Samsung's reachability technique: by default, it downscales the UI to 2/3 of its original size and moves it to the lower right corner (Fig. 6.2, center). Unlike Samsung's trigger that is either a swipe from the corner or a triple tap on the Home button, we chose iOS' double tap activation gesture: Swiping could have been confounding with BC's trigger, and pilot tests revealed that users found performing triple taps confusing. Since the iOS SDK does not notify about touch events on the Home button, users instead double-tapped above it on a pink virtual button on the touchscreen. The shrinking and movement animation duration was 265 ms.

MS lets the user control a stick that moves away from the user's finger. The stick automatically snaps to the next selectable target.

- **MagStick (MS)** [Roudaut et al., 2008] is triggered by force-touching on the screen using the same threshold as used for FR. To highlight a target, the user drags her thumb, which displays a line that grows into the opposite direction of where the thumb is moved. The line has two components of the same length, with the center located at where the thumb was initially placed. When the upper component gets within 5 mm of a target, it is highlighted, and the line snaps to its center as if magnetized. Lifting off the thumb selects the target.

BC lets the user swipe a virtual thumb extension from the bottom or sides of the smartphone screen.

- **BezelCursor (BC)** [W. H. A. Li et al., 2013] is triggered by swiping from the bezel towards the touchscreen. This displays a line that grows linearly by a factor of three in the direction of the thumb. The end of the line has a circular cursor that expands exponentially up to 7.3 mm depending on how fast the user swipes her thumb. This area cursor is equivalent to DynaSpot [Chapuis et al., 2009], and shrinks co-exponentially when swiping acceleration falls below 2 $\frac{mm}{s}$. When a target is intersected by the cursor, it is highlighted. When multiple targets are crossed, the target with the smallest distance from its center to the cursor location is chosen. Lifting off the thumb selects the target.

- **ForceRay (FR)** implements the interaction design described earlier. We set the force touch threshold to 1.33 units, which is significantly higher than an ordinary touch. We implemented the Quick Release mechanism for confirming the selection of a highlighted target as described in Chapter 5 using the default parameters. Pilot testing, however, revealed that these did not fit all users, leading to increased selection errors. We followed the approach presented in Chapter 5 to determine individual timing parameters by calibration trials upfront (M = 48–224 ms before complete thumb lift-off, SD = 32–48 ms). Pilot testing also optimized the CMC reference point placement to 3.2 mm away from the right and bottom of the screen; putting it exactly in the lower right corner would not allow users to comfortable move the ray to the right edge. For mapping the force input to the cursor position (C_{pos}) on the ray, we first used a linear transfer function as suggested by Stewart et al. [2010], but users tended to overshoot targets. Therefore, we designed a logarithmic transfer function whose slope decreased from 50% to 25% from force touch threshold to maximum sensible force: $C_{pos}(F_{rel}) = (\frac{2}{5}\ln(F_{rel} + \frac{3}{5}) + .0893) \cdot s + \delta$, where F_{rel} is the relative force ranging from force touch threshold to maximum sensible force, s denotes a scaling factor for the screen size, and δ represents the distance from the ray origin to where the cursor should start on the ray. For our setup, we set s = 1,852 pt and δ = 248 pt minus the dynamic distance from the thumb's touch location to the CMC reference point. Furthermore, to reduce ray and cursor jitter, we filtered touch location and force using the 1€ Filter [Casiez et al., 2012] (α = .1).

FR works as described above. As anchor point for the ray, we set the lower right corner of the smartphone that mimics the location of the thumb's CMC joint. We placed the anchor point to the right because all study participants were right-handed.

Targets

Targets were arranged on an invisible 6 × 10 grid (Fig. 6.2) across an area of 414 × 730 pt; each cell measured 69 × 73 pt. The bottom 414 × 6 pt of the screen was excluded to obtain cell sizes with whole numbers. We centered targets at the cells and added distractors to other cells. Like Karlson et al. [2007], distractors were not centered to avoid a regular-looking arrangement. Since reachability techniques are only meant to replace direct touch

The targets that participants had to acquire were arranged on an invisible grid. Within each cell, the center points of the targets were shifted.

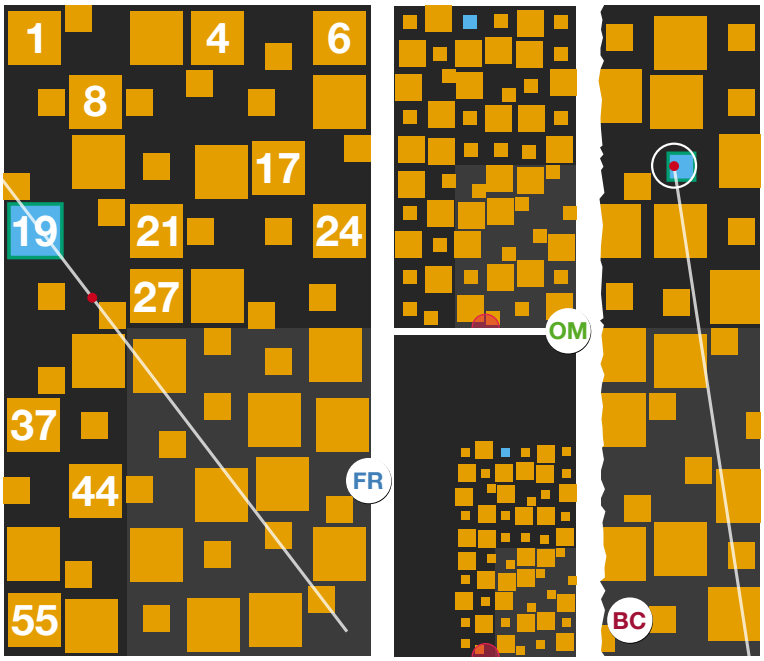


Figure 6.2: TARGETS (numbered) and distractors arranged on an invisible 6 × 10 grid. FR: layout for the large target SIZE condition. The user is aiming for the blue target that is crossed by the ForceRay. Currently, the target with the green border is selected since it is the next target on the ray beyond the cursor. OM (miniaturized): Target layout for the small target SIZE condition. Top: Full layout before double-tapping the virtual home button (pink). Bottom: UI downscaled by 2/3 after double-tapping the pink button. BC: The user is selecting the blue target with BezelCursor. The red dot visualizes the cursor position that is enlarged by the concentric white circle (DynaSpot area cursor [Chapuis et al., 2009]). Targets in the lower right area (brighter background) were not selectable, since here, targets are directly accessible by the thumb.

Targets in the lower right corner that were always reachable anyway, were not selectable.

input for targets that are out of reach, distractors close to the thumb (bottom right 4 × 5 cells) were not selectable. Participants were informed about this and a slight brighter background subtly visualized the exact area. For all techniques, lines were drawn in white, the target to select in blue, the cursor in red, and the border of a highlighted target in green. Hence, when the blue target was highlighted by the green border, the user should lift her thumb. After a target was selected, the next trial was shown

automatically after 500 ms. For OM, the UI was automatically shifted back right before showing the next trial.

Variables

Independent variables were Technique, Target, and their Size.

Independent Variables were Technique (DT, OM, MS, BC, and FR), Target, that split our twelve targets (Fig. 6.2) into two groups: targets 1, 4, 6, 19, 37, and 55 located at the Border of the screen vs. the remaining six targets rather located at the Center, and their corresponding Size (small: 30 × 30 pt (4.8 × 4.8 mm) and large: 60 × 60 pt (9.6 × 9.6 mm)). Size represented typical UI widget sizes, like the height of a button (30 pt) or an app icon (60 pt) on the iOS Home Screen.

each participant performed 280 trials, i.e., target selections using the five different Techniques.

We recorded 5 Technique × 12 targets × 2 Size × 2 repetitions = 240 trials per user. Technique was counter-balanced using a Latin Square. Targets were randomized. Size was also randomized, but we ensured that each user started testing a new Technique about 50% of the time with small vs. large targets first. When users were presented a new Technique, they familiarized themselves with the Technique before performing four test trials for the given target Size. They then selected the twelve targets, repeated two times, followed by the remaining Size for the current Technique, again starting with four test trials. After all 12 × 2 trials were performed, the next Technique was presented. Including test trials, each user did 280 trials in approximately 45 minutes.

Dependent variables were the task completion Time, the *Success*, i.e., whether the requested target was selected or not, the *Gesture Footprint* of the touch movements from the thumb, and the *Rotation* of the device around the x-, y-, and z-axis.

Dependent Variables were trial completion *Time* [ms], users' *Success* [0,1], i.e., whether they selected the correct target or not, and the *Gesture Footprint* caused by the touches on the screen to capture up to where users had to move their thumb. To quantify device motion and grip stability as done by Eardley et al. [2017] and Eardley et al. [2018b], *Rotation* captured device rotation [°] around x-, y-, and z-axis at 60 Hz. After each Technique, users were asked how much they agreed to (i) that they had to regularly change their device grip *before* acquiring a target, (ii) that they maintained a stable grip *while* selecting a target, and (iii) that the Technique was easy to apply on a 7-point Likert scale (7 = totally agree). At the end, users were asked to rank all Techniques by preference from highest (1) to lowest (5).

6.4.2 Results

For the analysis of *Rotation* data, we followed Eardley et al. [2017] and Eardley et al. [2018b] by summing up the absolute angles of device motion change around each axis.

Since we were interested in how users' performance is different depending on the Technique used, we will focus our analysis on this main effect and related interaction effects. We conducted a repeated-measures ANOVA on the log-transformed *Time* data. For the dichotomous *Success* data, we ran McNemar and Cochran's Q tests. For the analysis of *Rotation* data we followed Eardley et al. [2017] and Eardley et al. [2018b] by summing up the absolute angles of device motion change around each axis and ran repeated-measures ANOVAs on the log-transformed data. Likert scale data was compared using Friedman tests.

As to be expected, DT was the fastest Technique, followed by BC and FR. Only for FR, participants were faster at selecting targets at the *Border* targets compared to targets at the *Center*.

Technique had a significant main effect on ***Time*** ($F_{4,3565} = 348.95$, $p < .001$). Tukey HSD post hoc pairwise comparisons were all significant ($p < .001$) except between OM and MS (Fig. 6.3, left). As expected, users were fastest with DT (1,153 ms), followed by BC, for which they needed ≈ 324 ms longer. FR was the third fastest Technique, yet less than ≈ 200 ms slower than BC. OM and MS were close to 2,000 ms. There was also a Technique × Target interaction effect ($F_{4,3565} = 39.02$, $p < .001$). Figure 6.4 (top) list the Tukey HSD post hoc test results. For each Technique except FR, users needed significantly more time to select Border targets compared to Center targets. For FR, on the contrary, this was reversed.

Success was highest for BC (99%) and FR (96%). The small difference in *Success* between the two Techniques was not significant.

Technique had a significant main effect on ***Success*** ($Q(4) = 81.02$, $p < .001$, Fig. 6.3, right). Post hoc tests revealed that *Success* for BC was significantly higher compared to all other Techniques except FR. Furthermore, FR and DT yielded significantly higher *Success* than OM and MS. There was also a Technique × Size interaction effect ($Q(9) = 201.30$, $p < .001$). Figure 6.4 (bottom) shows the results from the post hoc tests. For small targets, BC yielded significantly higher *Success* than DT, OM, and MS, and FR had significantly higher *Success* than OM and DT. For large targets, MS had significantly lower *Success* than all other Techniques. Only for DT, there was a significant difference for *Success* comparing both Sizes: Small targets had 11% lower *Success* than large targets.

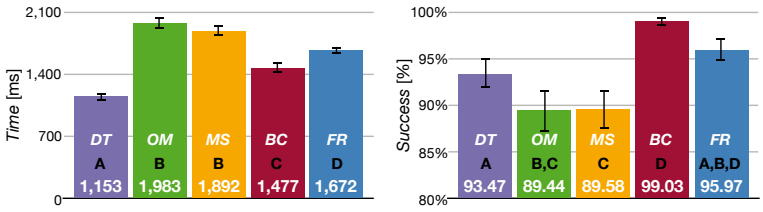


Figure 6.3: Study 1: *Time* [ms] (left) and *Success* [%] (right) by Technique. For each variable, pairs of levels that do not share a letter are significantly different (*Time*: all p <.001, *Success*: all p <.05). Whiskers denote 95% CI.

		DT			OM			MS			BC			FR		
		M	CI		*M*	CI		*M*	CI		*M*	CI		*M*	CI	
Size	Small	1,222	±65	A	2,074	±98	A	2,049	±93	A	1,574	±96	A	1,790	±61	A
	Large	1,084	±39	A	1,892	±72	A	1,735	±72	A	1,381	±56	A	1,554	±53	A
Target	Border	1,266	±67	A	2,148	±94	B	2,070	±91	B	1,552	±96	C	1,520	±44	C
	Center	1,039	±33	A	1,818	±75	B	1,714	±73	B	1,403	±57	C	1,824	±66	B

Time [ms]

		DT			OM			MS			BC			FR		
		M	CI		*M*	CI		*M*	CI		*M*	CI		*M*	CI	
Size	Small	88.06	±2.95	A	81.11	±3.71	B,D	88.89	±2.84	A,D,E	98.89	±.68	A,C	94.72	±1.87	C,E
	Large	99.17	±.55	A	97.78	±1.09	B	90.28	±2.65	B	99.17	±.55	B	97.22	±1.26	B
Target	Border	93.06	±2.20	A	86.11	±3.19	B	86.67	±3.13	B	98.89	±.68	A	98.33	±.90	A
	Center	93.89	±2.04	A,B	92.78	±2.25	A	92.5	±2.29	A	99.17	±.55	B	93.61	±2.09	A,B

Success [%]

Figure 6.4: Study 1: *Time* [ms] (top) and *Success* [%] (bottom) by Technique × Size and Technique × Target. Yellow cells denote significant differences within Technique (*Time* and *Success*: all p <.001). Pairs of levels that do not share a letter are significantly different across Technique (*Time*: all p <.001, *Success*: all p <.05). CI denotes 95% CI.

Technique had a significant main effect on ***Rotation*** around the *x-axis* ($F_{4,3566}$ = 995.81, p <.001), the *y-axis* ($F_{4,3566}$ = 778.61, p <.001), and the *z-axis* ($F_{4,3566}$ = 563.33, p <.001). For the *y-axis* and the *z-axis*, Tukey HSD post hoc tests revealed that all Techniques were significantly different from each other (Fig. 6.5). For the *x-axis*, post hoc tests revealed that differences between DT and OM and between BC and MS were non-significant. All other pairwise comparisons were significantly different (all: p <.001). In summary, for each angle, FR always caused the fewest device movement. There were also Technique × Target interaction effects for the *x-axis* ($F_{4,3566}$ = 4.30, p = .002) and for the *y-axis* ($F_{4,3566}$ = 4.52, p = .001). Figure 6.6 lists the results from the Tukey

FR always causes the smallest device rotation around the x-, y-, and z-axis, indicating that users held the device the most stable compared to all other Techniques.

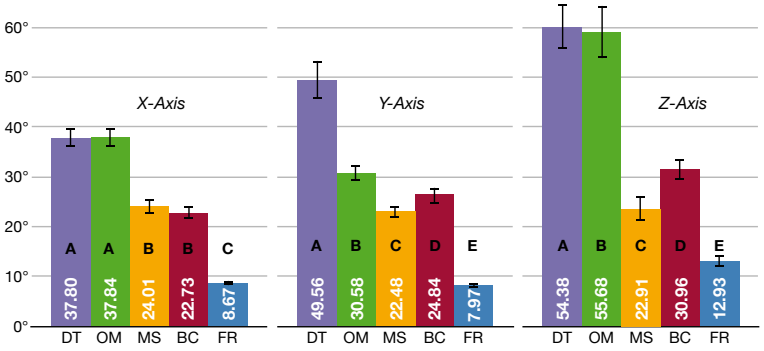


Figure 6.5: Study 1: *Rotation* [°] for the *x-axis* (left), the *y-axis* (middle), and the *z-axis* (right) by TECHNIQUE. For each variable, pairs of levels that do not share a letter are significantly different (all p <.001). Whiskers denote 95% CI. FR caused almost no device movement.

		DT			OM			MS			BC			FR		
		M	CI		*M*	CI		*M*	CI		*M*	CI		*M*	CI	
SIZE	Small	39.30	±2.82	A	38.52	±2.83	A	24.40	±1.87	A	23.64	±2.07	A	8.93	±0.61	A
	Large	36.29	±2.25	A	37.16	±2.63	A	23.63	±2.22	A	21.83	±1.48	A	8.41	±0.58	A
TARGET	Border	44.12	±2.94	A	41.73	±2.90	A	28.36	±2.34	B	25.54	±2.02	B	9.47	±0.61	C
	Center	31.48	±1.88	A	33.95	±2.49	A	19.67	±1.60	B	19.92	±1.49	B	7.87	±0.57	C
SIZE	Small	49.20	±5.22	A	30.81	±2.16	A	23.44	±1.74	A	27.62	±2.30	A	8.39	±0.56	A
	Large	49.56	±5.8	A	30.58	±2.05	A	22.48	±1.77	A	24.84	±1.76	A	7.97	±0.64	A
TARGET	Border	57.84	±6.40	A	33.80	±2.46	B	26.60	±1.90	C,	28.61	±2.32	C	8.74	±0.57	E
	Center	40.92	±4.29	A	27.58	±1.60	B	19.33	±1.50	C	23.85	±1.71	D	7.62	±0.63	E
SIZE	Small	65.80	±6.65	A	62.22	±8.18	A	24.36	±3.70	A	31.90	±2.29	A	13.08	±1.34	A
	Large	54.38	±5.63	A	55.68	±6.56	A	22.91	±2.91	A	30.96	±2.97	A	12.93	±1.62	A
TARGET	Border	69.11	±6.92	A	65.19	±8.05	A	27.44	±3.07	A	34.12	±2.60	A	14.41	±1.65	A
	Center	51.07	±5.20	A	52.71	±6.67	A	19.83	±3.53	A	28.74	±2.68	A	11.60	±1.28	A

X-Axis / Y-Axis / Z-Axis

Figure 6.6: Study 1: *Rotation* [°] for the *x-axis* (top), the *y-axis* (middle), and the *z-axis* (bottom) by TECHNIQUE × SIZE (top) and TECHNIQUE × TARGET (bottom). Yellow cells denote significant differences within TECHNIQUE (all p <.001). Pairs of levels that do not share a letter are significantly different across TECHNIQUE (all p <.01). No significant differences were found for the *z-axis*. CI denotes 95% CI. SIZE had no effect, but Border targets caused more *Rotation* than Center targets.

HSD post hoc tests. In general, for each TECHNIQUE, acquiring targets at the Border resulted in more device movement around

both, the *x-axis* and the *y-axis*, compared to targets located towards the Center.

FR caused the smallest *Gesture Footprint*.

Figure 6.7 shows the ***Gesture Footprint*** generated by each Technique. FR caused the smallest and most coherent footprint, and touches stayed within the thumb's comfortable reach, following natural rotation around the CMC joint.

Participants stated that DT required to most grip changes during the trials. They preferred BC most and FR second among the tested Techniques to solve this problem.

Figure 6.8 shows the mean and 95% CI for the questionnaire data. ***Grip change*** had a significant effect on Technique ($\chi^2(4) = 36.19$, $p < .001$). Regarding post hoc tests, users stated significantly more grip changes for DT compared to all other Techniques (all: $p < .05$) The same trend was found for ***grip stability*** ($\chi^2(4) = 36.89$, $p < .001$, post hoc tests: all: $p < .05$). For ***ease of use*** ($\chi^2(4) = 18.27$, $p = .001$), users found BC significantly easier to apply than MS ($p = .001$). Technique had also a significant effect on users' ***ranking*** ($\chi^2(4) = 21.55$, $p < .001$). Overall, participants preferred BC most, followed by FR, OM, MS, and DT, with BC being significantly preferred over all other Techniques (all: $p < .05$) except FR.

6.4.3 Discussion

As expected, DT was fastest but had a fairly low *Success* rate because users had to frequently change their grip to be able to reach all targets.

Overall, DT was fastest but achieved low *Success* for small targets, matching previous findings (e.g., from Karlson et al. [2007] and Karlson et al. [2008a]). DT also caused the strongest device motion, since especially at extremely far positions, like the upper and lower left corner, users had to change their grip, for which some participants almost accidentally dropped the phone, matching findings from Eardley et al. [2017]. This was why users preferred DT the least. Surprisingly, OM caused the second highest device motion.

OM's animation duration takes significant time for completing a trial and small targets become too small to hit precisely.

Time for OM was highest, reaching almost 2,000 ms. This could be due to the double tap trigger, which, unlike BC and FR, does not contribute to the target selection process, and due to the 265 ms animation time that helps the user understand how the UI is transformed. Both take additional time. *Success* for OM was also low, especially for small targets (Fig. 6.4, bottom: 81.11%): Shrinking them makes them too hard to hit precisely due to the thumb's fat finger problem [Siek et al., 2005]. A solution like

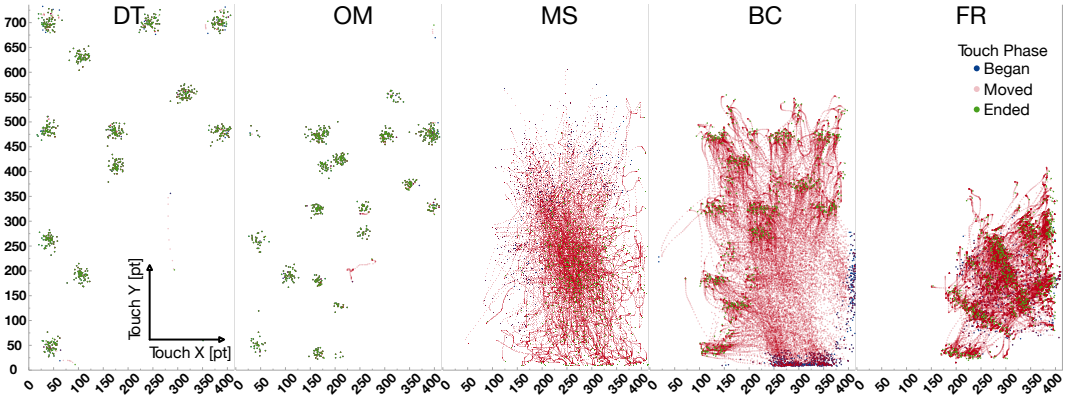


Figure 6.7: Study 1: *Gesture Footprint* by Technique. Blue dots mark where users started placing their thumb, red dots represent dragging, and green dots indicate where users lifted their thumb. FR had the smallest coherent footprint and followed ergonomic thumb movement.

AnglePose [Rogers et al., 2011] that captures the finger's orientation, could help gaining touch precision.

With MS, participants must carefully plan where to place their thumb because otherwise they might not be able to reach the distant target.

MS had the highest task completion time and lowest *Success*. Participants found MS more demanding than other techniques, because it required good planning upfront: To reach the top left corner, e.g., the user must place her thumb at least above the center of the screen, so that dragging it to the lower right corner will move the cursor far enough in the opposite direction. When the thumb is placed improperly, it is not possible to correct this within a trial, which explains MS' low *Success*, for both, large and small targets. Using the model from Bergstrom-Lehtovirta et al. [2014], we found that for most of our users, the center of the 5.5" screen is not reachable without uncomfortably stretching the thumb or changing the device grip.

BC is fastest, but for targets in the upper region of the screen, participants started to tilt the device to ease swiping.

Among the reachability techniques, BC was fastest, which was also independent from target Size. While users ranked BC best, they remarked that targets at the top were more difficult to reach due to swiping long distances upwards. To ease this, users tilted the device (Fig. 6.6) towards the thumb, for which they sometimes even had to reposition their grip.

FR's slightly lower *Success* than BS could be due to generic

Although non-significant, *Success* for FR was 3% lower than for BC. The Quick Release selection mechanism used in FR could be one explanation, since a lift-off based on force is a less definite

	"I had to change my grip regularly."			*"I maintained a stable grip while selecting."*			*"The technique was easy to apply."*			Rank 1: Highest, 5: Lowest		
	M	**CI**		***M***	**CI**		***M***	**CI**		***M***	**CI**	
DT	6.20	±.87	A	2.13	±.65	A	5.20	±.92	A,B	4.00	±.66	B
OM	3.27	±1.08	B	4.53	±1.00	B	5.33	±.83	A,B	3.13	±.65	B
MS	2.27	±.80	B	5.60	±.78	B	4.53	±.91	A	3.47	±.63	B
BC	2.73	±.87	B	4.80	±.92	B	6.47	±.29	B	1.47	±.47	A
FR	1.33	±.27	B	6.40	±.28	B	5.33	±.80	A,B	2.93	±.79	A,B

Figure 6.8: Study 1: Means and 95% CI for Likert scale responses (1: totally disagree, 7: totally agree) and ranking data (right) from the questionnaire. For each statement, pairs of levels that do not share a letter are significantly different. BC was preferred over FR, but not significantly more.

event than a lift-off only considering touch information, as used for BC. Unfamiliarity with force control could also explain the lower *Success*: as we will see in Chapter 7, users' performance for force control significantly increases with training. Our participants reported that selecting close and far targets with FR was fundamentally easier compared to other targets because then they only had to apply minimum or maximum force. Some developed a strategy: Independent of the target location at the border, they first applied maximum force to create a ray with the cursor positioned at its end and then, while maintaining this force, moved the ray onto the target.

timings for the QR mechanism and unfamiliarity with force control.

Interestingly, some participants remarked that they found FR's continuous cursor rather a disadvantage and that the discrete green indicator highlighting the preselected target was sufficient. Despite the 1€ Filter [Casiez et al., 2012], these users were worried when the cursor would move along the ray with subtle changes in force. Furthermore, they remarked that they were concentrating too much on that cursor, trying to push it towards the target's center instead of stopping when the green highlight matched the desired target.

Some participants found the continuous visual updates of the cursor position on the ray disturbing.

Taking all data from Study 1 into account, we found BC as the fastest and most accurate reachability technique. However, BC's *Success* was not significantly better than FR's, and FR caused the least and almost no device motion that was significantly different from all tested techniques, and it allowed users to select all targets without leaving the thumb's comfortable region. Yet, users found controlling their force the main challenge. We wondered whether users could improve their performance by training. We

We wondered whether training could help participants improve *Time* and *Success* for FR. We therefore conducted a second user study.

therefore conducted a second user study with trained users testing the two most promising and preferred techniques: BC and FR.

6.5 Study 2: Trained User Performance

Six participants trained FR and BC for three days, four 10-15 minutes sessions, per day.

We asked six people (23–34 years, M = 25.50, SD = 3.33; two female; all right-handed; thumb length: M = 68.33 mm, SD = 4.37 mm; phone screen size: M = 5.33", SD = .51") to train FR for three consecutive days, four sessions per day. To train in the same consistent environment, users were asked to come to our lab. For fair comparison, they also trained BC.

6.5.1 Apparatus and Task

The task was the same as in the first user study, but this time, participants had to select all of the 40 available targets that were randomly shifted off their cell's center. Before participants started with their training, we determined their individual QR timings.

We slightly modified the application from Study 1: We used the same grid for laying out targets, but this time users selected all 40 targets in random order. Half of the targets were of small Size (4.8 mm), the other half of large Size (9.6 mm). Targets were shifted from the center of their grid cells at random. All targets were repeated twice per Technique. Hence, in each training session, a user performed 2 Techniques × 40 Targets × 2 repetitions = 160 trials. Based on the feedback from Study 1, we disabled the continuous cursor for FR after 50% of the training sessions. Otherwise, dependent and independent variables were the same as in Study 1. Again, users performed a calibration task to determine individual Quick Release timings for FR before the first session. Sessions alternated between starting with BC and FR.

After three days of training, participants did a final session at our lab, using a similar setup as in the first study. Prior to this session, we again determined their individual QR timings.

Three days later, our users performed a final session, preceded by one more training session as a warm-up exercise and a renewed Quick Release calibration, since training could have affected the individual timings. The final session followed the original design from Study 1, but excluded DT, OM, and MS. Afterwards, users filled in the same questionnaire used in Study 1. In addition, we asked them how much they agreed to that even further training would allow them to (i) select targets faster, (ii) hit more correct targets, and (iii) maintain a more stable device grip. We also

asked users how much they agreed to have felt fatigue in arm, hand, or fingers. All responses were based on a 7-point Likert scale.

6.5.2 Results

Figure 6.9 (top) shows how participants' *Time* decreased over the twelve training sessions for both BC and FR. After twelve sessions, trained FR was as fast as untrained BC. Yet, after training, BC was still 250 ms faster than FR. *Success* for BC yield a constant 98–99% *Success*, but for FR the 96–97% *Success* decreased to 94–95% for the last four training sessions.

Over the twelve training sessions, *Time* decreased for both, BC and FR, but BC was still a 250 ms faster than FR.

For analysis, we focus on the main effects from the final session conducted after training.

TECHNIQUE had a significant main effect on ***Time*** ($F_{1,563}$ = 13.24, p <.001): BC (1,248 ms) was significantly faster than FR (1,372 ms), but the difference was small (Fig. 6.10, top).

BC was significantly faster than FR.

TECHNIQUE had a significant main effect on ***Success*** ($\chi^2(1)$ = 7.69, p = .003, Fig. 6.10, bottom): *Success* for BC (99.65%) was significantly higher than for FR (95.83%).

BC had significantly higher *Success* than FR.

TECHNIQUE had a significant main effect on ***Rotation*** around the *x-axis* ($F_{1,563}$ = 1225.14), the *y-axis* ($F_{1,563}$ = 1581.83), and the *z-axis* ($F_{1,563}$ = 871.92, all p <.001). For each axis, *Rotation* was significantly lower for FR than for BC (Fig. 6.11, left).

Rotation for FR was for each axis significantly lower than for BC.

Figure 6.12 shows the results from the questionnaire: Users stated to have done significantly more ***grip change***s with BC compared to FR ($\chi^2(3)$ = 16.47, p = .001). Responses for ***grip stability*** and ***ease of use*** showed no significances. Also, users disagreed to have felt ***fatigue*** for both BC and FR. The ***Gesture Footprint*** for BC and FR was similar to Study 1.

Participants reported to have performed more grip changes for BC than for FR.

6.5.3 Discussion

Overall, training sped up both, BC and FR. Whereas in Study 1, users moved the ray and the cursor for FR sequentially, we

With training, participants became

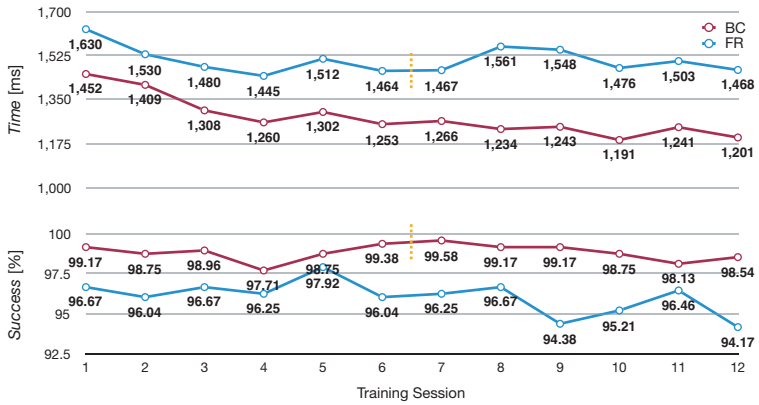


Figure 6.9: Study 2: *Time* [ms] (top) and *Success* [%] (bottom) for all training sessions. Users became faster over all sessions for both, BC and FR. *Success* for BC was ≈ 99% for each session and for FR ≈ 97% for sessions 1–8, but then decreased due to the Quick Release calibration not fitting trained performance anymore. The dashed line indicates when FR's red cursor was turned off, showing no effect on FR performance.

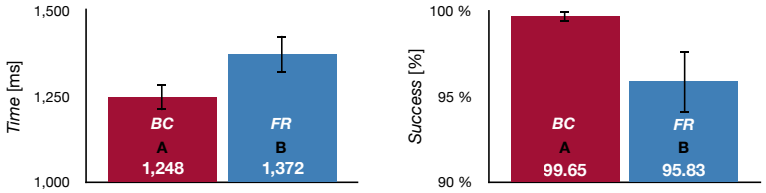


Figure 6.10: Study 2: *Time* [ms] (left) and *Success* [%] (right) by Technique. Pairs of levels that do not share a letter are significantly different (*Time*: p <.01, *Success*: p <.05). Whiskers denote 95% CI. While BC was overall faster than FR, users were equally fast for Border targets (cf. Discussion).

confident in merging ray and cursor control for FR simultaneously instead of sequentially.

observed during training that they became confident in merging both steps. Users were now 15.5% faster for BC and 18.0% faster for FR after training. The significant difference between BC and FR could be explained by a Technique × Target interaction effect ($F_{1,563}$ = 32.76, p <.001): While for BC, Target had no effect on *Time*, users selected Center targets significantly slower (1,508 ms) compared to Border targets (1,235 ms) when using FR (post hoc p < .001). Compared to CornerSpace and BezelSpace [Yu et al., 2013] (cf. Related Work) that target reach-

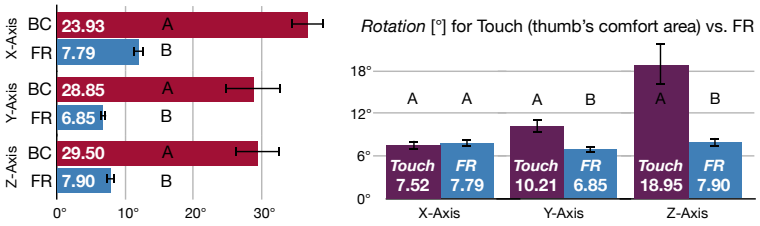


Figure 6.11: Study 2: *Rotation* [°] around the *x-axis*, the *y-axis*, and the *z-axis*. Left: BC vs. FR. Right: Touching within the thumb's comfortable area vs. FR. For each axis, pairs of levels that do not share a letter are significantly different (all p <.001). Whiskers denote 95% CI. FR had the least device movement.

	"I had to change my grip regularly."			*"I maintained a stable grip while selecting."*			*"The technique was easy to apply."*			*"I felt fatigue in my arm, hand, or fingers."*		
	M	CI		*M*	CI		*M*	CI		*M*	CI	
BC	3.83	±1.40	A	6.50	±.57	A	4.17	±1.92	A	2.33	±1.44	A
FR	1.17	±.43	B	5.83	±.79	A	6.67	±.54	A	2.00	±1.63	A

Figure 6.12: Study 2: Means and 95% CI for Likert scale data (1: totally disagree, 7: totally agree) from the questionnaire. For each statement, pairs of levels that do not share a letter are significantly different. Unlike FR, BC required grip changes.

ability at smartphone bezel and corners FR was faster than both techniques. While our study was not exactly the same, Yu et al. [2013] also studied target selection on 5.5" devices.

Disabling the continuous cursor visualization did not influence performance.

Comparing the two FR sessions between which the red cursor was turned off, there was neither a difference for *Time* ($F_{1,953}$ = .29, p = .59, ns.), nor for *Success* ($\chi^2(1)$ = .00, p = 1.00, ns.), hence no influence on FR performance.

With training, participants became faster in performing the QR gesture.

The *Success* rates for BC and FR were almost unchanged compared to Study 1, yet, this time, significantly different, which could be explained by a Technique × Target interaction effect ($Q(3)$ = 15.00, p = .002): For Center targets, *Success* for FR was significantly lower (93.75%) compared to BC (99.31%) (post hoc p <.05). The slightly decreased *Success* for FR towards the last training sessions could be explained by the training effect: With increasing confidence, users tended to lift off their thumb faster, which affected the Quick Release mechanism. In fact, calibration parameters shrunk from 32–240 ms to 32–192 ms.

Participants think that further training could help to master FR even better.

When users were asked regarding their agreement to that further training would help them becoming (a) faster and (b) more accurate in selecting targets, users slightly agreed to this for FR (a: M = 5.33, b: M = 5.17) , but were rather undecided regarding BC (a: M = 3.33, b: M = 3.33).

Participants still tilted the device for reaching targets in the upper area of the screen when using BC.

Training did not reduce BC's device motion, but for FR, z-axis rotation decreased by 39% compared to Study 1. Although, in theory, swiping from the sides could have helped to compensate tilting the device to reach targets near the top edge when using BC, users rarely made use of this. Three participants remarked that swiping from the sides was more difficult than from the bottom since the sliding space at the side is smaller than at the bottom due to the height of the Home button. This space, however, will disappear on bezel-free devices, like iPhone Xs. Protection cases with protruding edges make swiping from the side also less comfortable. Furthermore, smartphone operating systems usually reserve bezel-swipes to let the user navigate back, or access notifications or settings. Although these mappings are in flux, e.g., iPhone X moved the control center shortcut to the top right, it breaks user practice and makes shortcuts harder to access since they are now out of one-handed reach. Accessing also BC via these swipe gestures requires removing all established shortcuts from the bottom and the sides to the top since BC determines the ray direction based on the user's initial swipe touch points. This is likely to result in a gesture overload at the top.

FR causes even less device rotation for distant targets compared to classic direct touch input for targets that are easily reachable by the thumb.

To compare usual device motion caused by tapping within the thumb's comfort region vs. device motion caused by FR, we re-recruited the six participants to let them tap targets in the lower right corner of the touchscreen (highlighted area in Fig. 6.2). A repeated-measures ANOVA on the log-transformed *Rotation* data revealed no difference for the *x-axis* between the two techniques ($F_{1,573}$ = 1.86, p = .17, ns.), but for the *y-axis* ($F_{1,573}$ = 28.04, p <.001) and the *z-axis* ($F_{1,573}$ = 32.44, p <.001) FR caused even significantly less motion (Fig. 6.11, right).

FR is beneficial for quickly reaching targets at edges and corners and when device and grip stability are vital.

In summary, FR is especially beneficial for selecting far targets at screen edges and corners, like the iOS back button, navigation bars, or slide-out menus. FR has also shown to cause the least device motion compared to BC and tapping in the thumb's comfort area, removing the need for re-grasping the device, and likely reducing potential device drops.

6.6 Guidelines

We present guidelines for the tested Techniques.

Based on what we have learned from Study 1 (S1) and Study 2 (S2), we give recommendations for the tested techniques for different criteria and contexts. Figure 6.13 visualizes a decision tree for these recommendations.

DT is still best when selection speed is ultimately critical.

When selection speed is ultimately critical, e.g., in games, DT is the technique of choice, despite grip changes that users have to perform for far targets (S1).

For large enough targets, OM is a good choice.

When direct touch input is important and targets are ≥ 60 pt, OM is a good compromise to also satisfy reachability for one-handed touchscreen use.

BC provides a good speed-accuracy trade-off for targets that are out of reach.

For a good speed-accuracy trade-off, BC is recommended (S1, S2). When the UI has many targets located at the center, yet beyond the thumb's reach, BC is fast and accurate and causes moderate device motion, likely without the need for a grip change. However, for targets located near and at the upper edge, such as menus or the iOS back button, users tend to tilt the device towards them to reach the target and often re-grasp the device to ease this.

For top- and side-located menu and navigation buttons, FR is a good choice and also provides excellent device and grip stability, e.g., for augmented reality apps.

For such targets, and, in general, when device and grip stability are important, FR is recommended (S1, S2). For example, apps that use the smartphone camera, such as apps for taking photos and videos, scanning documents, or augmented reality games, it is important to keep the camera focused at a static viewpoint. Also, when ergonomics are key, FR is most beneficial, since the user's thumb stays within its comfort region and rotates naturally around the CMC joint.

6.7 Limitations and Future Work

FR is dependent on the user's handedness and interferes with existing force touch interaction.

FR performance is affected by handedness: For targets located at the right screen edge, right-handed users cannot exploit target selection via maximum possible force, except for the upper right corner target, since the ray will cross all of these targets. This is why we did not consider target 24 a border target in the analysis.

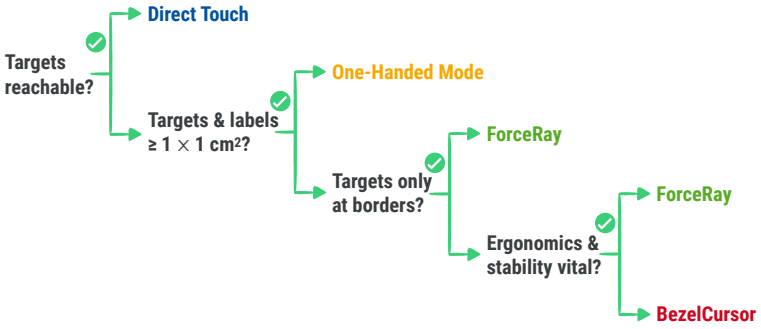


Figure 6.13: A decision tree for picking the appropriate reachability technique among the ones tested in this work. If targets are easily reachable, Direct Touch is the best choice since users acquire targets quickly with ease. If targets are out of reach but still reasonably large, the One-Handed Mode is a good option. With practice, the animation duration that shows the user where targets are moving could be reduced to speed up target selection. If targets are mainly located at the borders of the handheld touchscreen, such as menu buttons and navigation bars, ForceRay is a good choice. Also, if targets are located elsewhere but ergonomics for the thumb and grip stability are vital, ForceRay is the best technique for selecting targets. In all other cases, using BezelCursor should be considered.

For left-handed users, targets at the left screen edge are affected. Also, due to the polar coordinate system of the ray casting in FR, distant targets require more precise movement the smaller such targets are. In addition, FR sacrifices the benefit of direct manipulation (like BC) that both DT and OM share. Moreover, FR interferes with existing force input techniques within the thumb's comfortable region, such as force-touching an app icon in iOS. Using a different trigger, e.g., a subtle force touch that quickly drops it without lifting the finger off the screen ('Force Pulse', cf. Chapter 7), could mitigate this problem.

So far, FR only supports tapping of distant targets. A future extension of FR could tackle distant dragging and swiping.

Also, FR is only designed for targets responding to taps, but could be extended as follows: To discriminate taps from gestures, the user could dwell for 1 s on the distant target, and then reset the force by dropping it, yet without lifting the thumb off the screen: If the target responds to force, the user just presses. Swipes, instead, could be issued by Force Pulses, and scroll gestures could be issued by rolling the thumb left and right as discussed in

Chapter 7. Apart from testing this, we also plan a longterm study in which users control existing apps with FR while being in motion: On the one hand, walking negatively affects force control [Wilson et al., 2011]; on the other hand, FR's grip stability could then finally lead to a better performance compared to other reachability techniques. Finally, we would like to investigate whether a hybrid of BC and FR could combine the speed and stability benefits of both techniques: BC would be used as trigger and for generally aiming at out-of-reach targets. However, if a target is not comfortably reachable by BC anymore or causes significant device motion, the user could continue to extend the cursor using FR.

6.8 Summary and Conclusion

We designed a new interaction technique, called *ForceRay* (FR), that extends thumb reach via force input to enable selection of out-of-reach targets on handheld touchscreens with a steady device grip. To select a target that cannot be reached easily with the thumb, the user applies a force touch to cast a virtual ray in the direction of the target. By increasing her force, she moves a cursor along the ray until reaching the target, then lifts her thumb quickly to confirm the selection. We conducted two user studies to validate FR: Study 1 tested FR against existing reachability solutions. Among all, FR significantly caused the least device motion, removing users' concerns about device drops. Yet, FR was 195 ms slower than the fastest reachability technique, BezelCursor (BC). Study 2 showed that an hour of training sped up both BC and FR, and that both are equally fast for targets at the screen border.

We designed a new interaction technique, called *ForceRay* (FR) that solves reachability issues for handheld devices via force input with the thumb. In two user studies we showed that FR allows to reach all targets while the user can maintain a steady device grip.

With ForceRay, we enable users to make efficient of the touchscreen for input, since all targets are reachable with a single finger with ergonomic comfort. Besides making efficient use of input, force input can also contribute to making efficient use of output on the handheld touchscreen. Therefore, in the next chapter, we present the Force Picker, an interaction technique that

When more than only a few items need to be selected via force input, an absolute force-to-value mapping is inefficient.

Next, we will look at how rate-based force control makes value input faster.

enables efficient input of values from large range using a single finger. We will show how this technique uses force to save both, input and output space on the handheld touchscreen, such that the limited screen size can be used more efficiently.

7

Making Value Selection on Handheld Devices Efficient via Force Touch Input

→ Summary

Picking values from long ordered lists, such as when setting a date or time, is a common task on smartphones. However, the system pickers and tables used for this require significant screen space for spinning and dragging, covering other information or pushing it off-screen. The Force Picker reduces this footprint by letting users increase and decrease values over a wide range using force touch for rate-based control. However, changing input direction this way is difficult. We propose three techniques to address this. With our best candidate, Thumb-Roll, the Force Picker lets untrained users achieve similar accuracy as a standard picker, albeit less quickly. Shrinking it to a single table row, 20% of the iOS picker height, slightly affects completion time, but not accuracy. Intriguingly, after 70 minutes of training, users were significantly faster with this minimized Thumb-Roll Picker compared to the standard picker, at the same accuracy and only 6% of the gesture footprint. We close with application examples.

Publications: The work presented in this chapter has been done in collaboration with Simon Voelker, Andreas Link, and Jan Borchers. The author of this thesis developed the research idea and relevant research questions, including the motivation of the work. Furthermore, he has designed all experiments and analyzed their data. Part of this work has been published as a Master's Thesis in 2017 [Link, 2017]. Most of this work has been published as paper in the Proceedings of ACM CHI 2018 [Corsten et al., 2018]. The author of this thesis is the main author of the paper. Most sections in this chapter are taken from the paper publication.

7.1 Motivation

So far, we controlled only a few values via force input, i.e., usually not more than ten different items, using the positional control mechanism (PC).

The previous presented technique, ForceRay, used positional control for selecting a target. Positional control has the benefit that the user can quickly adjust for over- and undershoots by simply reducing or increasing force. However, humans can only control up to ten different targets reliably using this control mechanism. When interacting with handheld touchscreens, users often need to select one out of many targets, e.g., when picking a value from an ordered list to set the date and time of an appointment.

Typical touch-based widgets for selecting many values take up significant screen space on handheld devices.

However, touch-based UIs for such tasks take up significant screen space. For example, Apple's default system "picker" occupies 3.3 cm, or 30%, of an iPhone 8's screen height. On a Samsung Galaxy S5 running Android, it is 33%. This space is required not just to display the widget *(display footprint)*, but also for the swipe gesture to spin the picker's wheels *(gesture footprint)*, as shown in Figure 7.1 (left). Tables are similar: with many items, such as in the iOS *Settings* app, they may occupy the entire screen, and still require scrolling similar to the picker. Slide-in keyboards take up similar space.

Since such touch-based widgets occupy significant screen space, contextual information must be temporarily pushed off screen.

When these UI elements appear, other content usually disappears: The date picker in the iOS Calendar app, for example, expands from a single row with the date to the equivalent of five rows when activated, pushing content below it down and often off-screen. This can make contextual information needed to pick a value, such as the end time for a meeting, suddenly disappear. Horizontal sliders are another alternative for value input on smartphones that is smaller in height, but they are not recommended for setting precise values [Nielsen Norman Group, 2015].

Current alternative input techniques that require minimal screen space, such as speech input, are often impractical and inefficient for value selection.

There are alternative interaction techniques that need minimal screen space. Of these, however, speech is socially awkward and time-consuming, tilt sensing makes screens hard to read at an angle and is difficult to use while walking [B. L. Harrison et al., 1998; Hinckley et al., 1999; Rahman et al., 2009], and remapping existing physical controls like volume buttons leads to inconsistent behavior across apps. Bendable interfaces [Schwesig et al., 2004] require two hands for good control, and squeezable de-

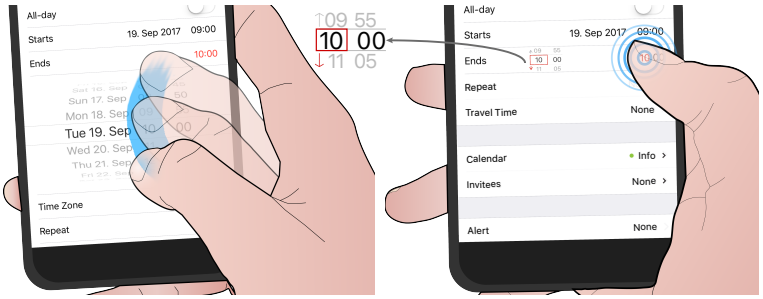


Figure 7.1: Entering a time on a smartphone. Left: Setting the hour with a picker requires significant screen space in height for dragging and spinning the wheel. Right: Selecting the hour by applying force on the hour digits of the underlying label fades in the *Force Picker*. Rolling the thumb to the left scrolls through the hours—the harder the user presses, the faster. Rolling the thumb to the right scrolls in the opposite direction. Lifting the thumb off the screen sets the value, and the Force Picker disappears. Compared to dragging and spinning, the Force Picker is more compact, so that contextual information is never pushed off-screen.

vices, like HTC's U11, can only sense two different states. All of these require an extra tap to specify where on the screen the input should go. Exploiting simple touch duration instead only allows for changing values at a constant rate, which takes time.

Using a rate-based control mechanism (RC), that maps force input to the velocity of scrolling through the set of values, force input can be a powerful technique to select values without occupying much screen space.

A more powerful option is to use the force-sensitive touchscreen in recent smartphones, like Apple's iPhones since the 6s or Huawei's Mate S, to add force as a third dimension to every touch input. There are two established control mechanisms to map such input to a selected value: *Positional Control (PC)* maps a particular force level directly to a particular value—the harder the user presses, the higher the selected value. As mentioned before, beyond around ten discrete values, however, this technique becomes infeasible [McLachlan et al., 2014; Wilson et al., 2010]. *Rate-Based Control (RC)*, by contrast, maps force to velocity: the harder the user presses, the faster the selected value increases. This allows for efficient selection from larger value ranges, such as dates, times, or percentages.

Unlike PC, RC does not support for

However, using force for RC has one major drawback: If exerting force increases the currently displayed value at varying speeds,

bidirectional value control, e.g., to quickly correct over- and undershoots.

how does the user *decrease* it? One solution are two buttons, one per direction, but this means that the displayed value can no longer serve as the input area itself, since the user needs to pick one of the buttons. This requires adding space for two permanent buttons next to the displayed value. Another alternative is the wrap-around technique: if the user scrolls past the highest value, it wraps around to the lowest value. However, this makes correcting overshoots tedious.

We challenged this problem with our *Force Picker* widget.

Instead, our *Force Picker* allows for bidirectional RC with force touch to select a value, using a technique to reverse direction that requires no more space than the initial touch.

In summary, we make the following contributions:

- the *Force Picker*, a force-based input technique for value selection that outperforms a standard picker on smartphones with trained users, at a fraction of the size;
- a quantitative study of three techniques for a Force Picker to reverse direction in-place;
- a quantitative study examining the effect of different display sizes for the Force Picker.

7.2 Related Work

In the Related Work section, we will focus on force-based input techniques.

There is a variety of techniques for entering values on smartphones, such as speech, sensing tilt, or remapping physical buttons, that do not use the touchscreen at all. However, as outlined in the Motivation, they exhibit other drawbacks, and still require the user to tap on the screen to specify which area the input should go to. We instead propose using force input as an alternative, and will focus on this area below.

We classify the Related Work regarding PC and RC mechanisms.

Related work on force input for selection tasks can be classified into the two groups of control mechanisms introduced above, Positional- and Rate-Based Control. They determine how force is used to navigate through the available options.

7.2.1 Positional Control (PC)

PC is an absolute mapping from force to value.

Positional Control (PC) maps force levels directly to selectable values: applying the same force always selects the same value. In terms of Card et al.'s seminal *Design Space of Input Devices* [S. K. Card et al., 1990], this is an "absolute" mapping. While typical force sensors are sampled with 10 bits of resolution for 1,024 possible raw sensor values [Shi et al., 2008], these raw sensor values are usually binned, mapping a range of them to one force level. A transfer function then maps force levels to values. This mapping is usually natural: the higher the force, the higher the value.

The transfer function determines how exactly force is mapped to values.

The transfer function in PC is often linear, as recommended by Stewart et al. [2010]. This satisfies the above criteria and is often used for menu selection. However, this approach begins to break down beyond around ten items: G. Ramos et al. [2004] showed that users can control up to six menu items reliably with a force-sensitive pen on a tablet. Mizobuchi et al. [2005] found similar results for menu control on a force-sensitive handheld device. Wilson et al. [2010] reported that users can control ten levels with their fingers on a handheld device at 85% accuracy with continuous visual feedback. McLachlan et al. [2014] added a force sensor to the bezel of a tablet; this enabled the hand holding the device to control a menu of ten items with 89% accuracy. Corsten et al. [2017a] added force sensors to the back of a smartphone, and found that users could control five levels of force reliably.

Cechanowicz et al. [2007] designed a fisheye transfer function that lets users select from up to 64 values using a force-sensitive mouse.

Navigating larger ranges with PC requires nonlinear transfer functions. Shi et al. [2008] reached 16 controllable levels using a fisheye transfer function with a force-sensitive mouse. Using a similar mouse, Cechanowicz et al. [2007] let users first tap on the force sensor multiple times to jump to a coarse level and then press to zoom in on that level at finer granularity, for up to 64 controllable levels. However, for bidirectional control, this technique requires a second force sensor.

In general, PC is inappropriate for selecting values from large ranges.

In summary, although bidirectional control within the footprint of a single finger is trivial with PC, the limited number of reliably selectable values makes PC inappropriate for setting values of larger ranges, such as times, dates, or percentages.

7.2.2 Rate-Based Control (RC)

With RC, one can browse value ranges of any size, but changing the direction requires a separate action.

Rate-Based Control (RC) maps force to the velocity of value change. The transfer function determines how fast values change depending on the force applied. Here, the usual natural mapping means that the harder the user presses, the faster values are browsed. Maintaining a particular force results in change at a constant speed; reducing the force applied slows down browsing. RC lets users browse value ranges of any size, but changing direction requires some separate input action.

A variety of RC-based techniques have been explored for both, desktop and handheld interaction.

Shi et al. [2009] used a force-sensitive mouse to rotate objects by mapping force to angular velocity. Users tapped the sensor to adjust by single degrees, and rotated counter-clockwise using a second force sensor. The system was faster to use than its PC counterpart. Wilson et al. [2011] compared RC to PC for controlling ten menu items on a smartphone using a single force sensor. For RC, they used wrap-around instead of bidirectional navigation. Using RC was faster and more accurate. Using one force sensor per direction, Ng et al. [2016] report similar results for menu control in a car. Spelmezan et al. [2013a] attached force and proximity sensors to the side of a smartphone for bidirectional continuous thumb scrolling. Their revised technique used two force sensors for bidirectional scrolling and zooming by using either the index and ring finger or thumb and palm [Spelmezan et al., 2013b]. While touch input was faster for small scrolling distances, RC prevailed for longer distances. Holman et al. [2013] attached two force sensors to the side of a smartphone to use with the fingers holding the phone while thumbing, and found that gestures augmenting the natural thumb touch worked best. Pelurson et al. [2016] attached a force sensor to a smartphone. Using RC to scroll large information spaces horizontally while changing direction by swiping on the touchscreen was faster and preferred over the default drag-and-flick interaction. Antoine et al. [2017] used RC on a force-sensitive iPhone to scroll a viewport vertically down while simultaneously dragging with the thumb. Users were faster and had fewer errors compared to baseline edge-scrolling.

Two examples that explore bidirectional RC are PreSense II

PreSense II [Rekimoto et al., 2006] and GraspZoom [Miyaki et al., 2009] describe bidirectional RC using force touch. Similar to our approach, they support immediate change of direction with a

minimal gesture footprint. In PreSense II, the user tilts her index finger up vertically to navigate in the opposite direction while force-touching on a desktop touchpad. In GraspZoom, swiping across the screen selects the direction of scrolling and zooming with RC, while a sensor on the back of the smartphone captures continuous force input. Both techniques, however, were not evaluated. Hence, it is not clear how fast and how accurate users perform with such techniques, especially for discrete value input, and whether such techniques let us minimize both gesture footprint and widget display footprint on the handheld touchscreen.

[Rekimoto et al., 2006] and GraspZoom [Miyaki et al., 2009].

7.3 Bidirectional Force Input Techniques

Below, we present three interaction techniques we designed for bidirectional value selection using force input that minimize the gesture footprint. Since we want to change values across a large range, e.g., 1–31 to set a date, all our techniques require RC, as our review of related work has illustrated.

We designed three RC-based techniques for value selection on force-sensitive smartphones.

To let standard touch input and RC using force touch coexist on the same widget without accidental activation, we classify any force input below a *resting threshold* of 20% (based on Apple's guidelines) of the force sensor range as an ordinary touch that does not start changing the displayed value. Between 20% and 80% of the sensor range, our force-to-value mapping iterates over values at a velocity depending on a linear transfer function, as used by Wilson et al. [2011]: The more force is applied, the faster the selected value changes. To efficiently cross large value distances, crossing 80% of the sensor range activates a *boosted* transfer function. We focus on one-handed interaction in portrait mode, using the thumb of the dominant hand. Lifting the thumb off the screen was chosen to confirm the currently selected item, removing the need for an extra tap or an additional confirmation button.

All three techniques (*Pulse*, *Press-Through*, and *Thumb-Roll*) are activated upon pressing with the thumb against the touchscreen and crossing a force threshold. This ensures that force input does not interfere with standard touch input.

By default, RC only supports changing values in one direction. Our designs (Fig. 7.2) address this shortcoming, each with its own benefits, within the footprint of a single force touch:

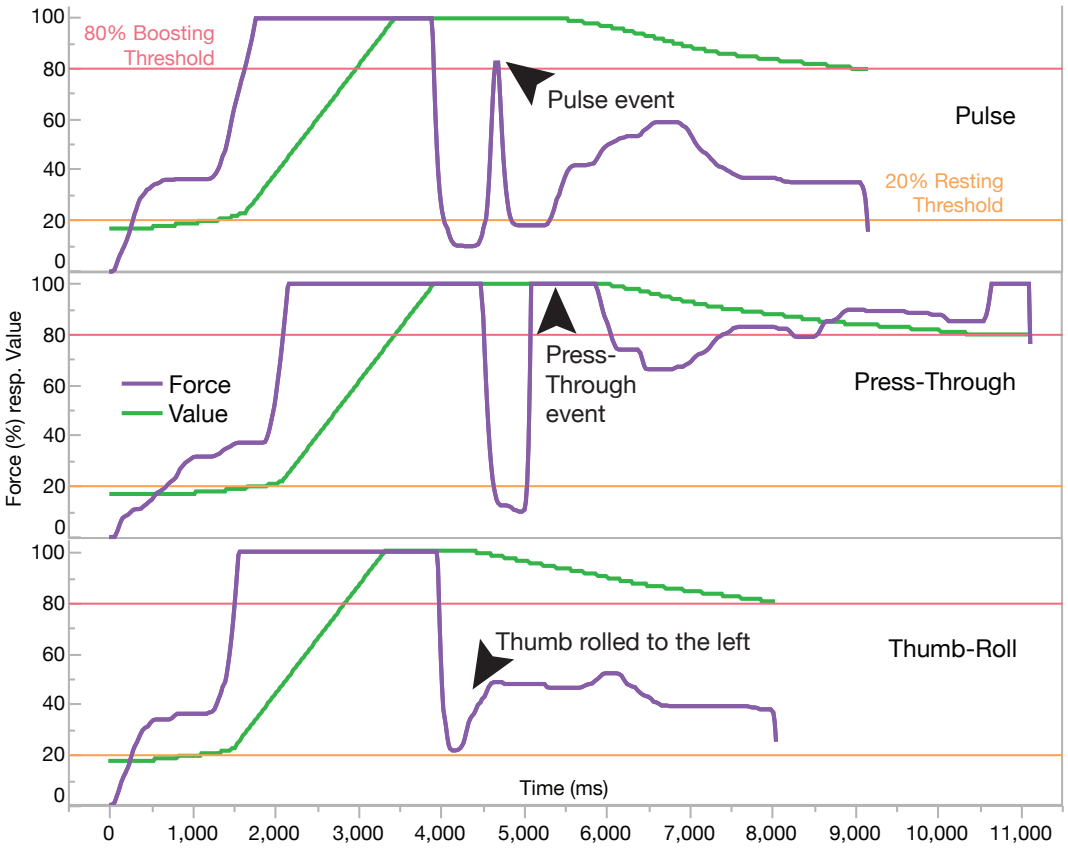


Figure 7.2: Force input and corresponding selected values for our three force techniques over time, when changing a value from 17 to 80 while overshooting to 100 in between. Note how values begin to decrease when the corresponding change of direction (black arrows) is detected.

Pulse changes the direction by quickly increasing and releasing force without lifting the thumb and then gradually increase force.

- **Pulse.** In *Pulse*, the user changes direction by quickly increasing force significantly above and back below the resting threshold. This technique has the advantage of being identical for both directions. To correct for overshoots in either direction, the user reduces force below the resting threshold, issues a Pulse, and increases force again to scroll towards the desired value.

Press-Through changes the direction by applying as much force as possible and then gradually reduce force.

- **Press-Through.** *Press-Through* maps force increase to value increase, and force decrease to value decrease: To reverse direction, the user quickly applies the maximum detectable force, starting from below the resting threshold. While she maintains that maximum force, value change pauses, like in the resting force zone. When she reduces

the force, the value starts decreasing, reversing the transfer function—the more she reduces the force, the faster the value decreases. Re-applying more force decelerates value decrease, up to the maximum detectable force, which pauses value changes. Within 40% to 20% of the force sensor spectrum, the boosted transfer function is applied. Below 20%, resting force is reached, which pauses value change and reverts back to the normal transfer function. To correct overshoots, the user quickly reduces force beneath the resting threshold, and then issues the Press-Through gesture to reverse direction. The advantage of Press-Through is its natural mapping of force increase/decrease to value increase/decrease.

- **Thumb-Roll.** In *Thumb-Roll*, the user gently rolls her thumb to the left to decrease, and to the right to increase values. The applied force sets the speed in either direction. To correct for overshoots, she rolls her thumb to the other side. This results in a natural decrease and therefore decelerated navigation while the thumb is rolling through its neutral, flat position. Since the thumb has to be rolled and maintained in that position while navigating, Thumb-Roll represents a temporary "quasi-mode" [Raskin, 2000] that ensures the user is always aware of the active direction from her thumb position. Like in Pulse, the transfer function is identical for both directions. Thumb-Roll is similar to a rocker switch, or two small force buttons right next to each other. Although rolling the thumb up and down would result in a more natural mapping for increasing and decreasing values, left-right roll is ergonomically easier to perform and leaves a much smaller footprint compared to rolling the thumb up and down. Roudaut et al. [2009] investigated thumb-roll gestures as input modifiers for traditional handheld touchscreens. Their technique MicroRolls gives users faster access to toolbars and menus on PDAs through four straight and two circular thumb roll gestures. However, that work did not explore force input.

Thumb-Roll changes the direction by rolling the thumb to the right (to increase) or to the left (to decrease) before gradually increasing force.

7.4 Study 1: Force Picker vs. System Picker

In Study 1, we wanted to compare how participants perform value selection with our Force Picker designs compared to a standard touch-based System Picker.

Having defined these candidate techniques to reverse direction for the Force Picker, we wanted to understand how they each compare to a standard picker that is controlled by dragging and spinning, in terms of speed, accuracy, and gesture footprint. 16 participants (21–31 years, M = 26.19, SD = 2.71, all right handed, five females) used a Force Picker with each candidate technique, and a standard picker as baseline, on a force-sensitive iPhone 6s Plus. Figure 7.3 shows our application to display instructions and capture data.

Picker Design

The visual design of the Force Pickers and the System Picker were similar to standard pickers used in iOS and Android.

For our baseline condition, we designed a **standard picker** similar to an iOS or Android system picker. We reserved the same 414 × 216 pt (65 × 36 mm, 30% screen height) space for the dragging and spinning footprint that Apple recommends to use for the iOS 10 system picker [Apple Inc., 2019a]. Note that the unit 'pt' does not refer to the unit measure used in typography, but relates to Apple's measurement for display points on iPhone 6/7/8 Plus (1 pt ≈ .16 mm). We displayed a rectangle with slightly narrower sides around this area, such that users could easily perceive the picker boundaries. Although the iOS system picker displays seven values at a time, of which the two upper and lower values are in a vertically compressed font (Fig. 7.4, left), we chose to display only three values at a time, like the Android system picker does (Fig. 7.4, right). This was done since we anticipated shrinking the display space for our Force Picker in Studies 2 and 3 (Fig. 7.3, right), which would be difficult for displaying seven values at a time, and we wanted to be able to do a fair comparison with our results across all studies.

To confirm selection using the standard picker, one must tap a dedicated 'select' button.

Dragging the current value down with the thumb anywhere on the picker area increased the value displayed and vice versa. We used the standard iOS 10 accelerations for dragging and spinning from the `UIScrollView` class. The target value was shown to the left so that it would not be covered by the thumb while dragging. The standard picker was placed at the bottom of the screen to make sure that it was easily accessible by the thumb (Fig. 7.3, center). Since liftoff happens frequently on standard pickers, it

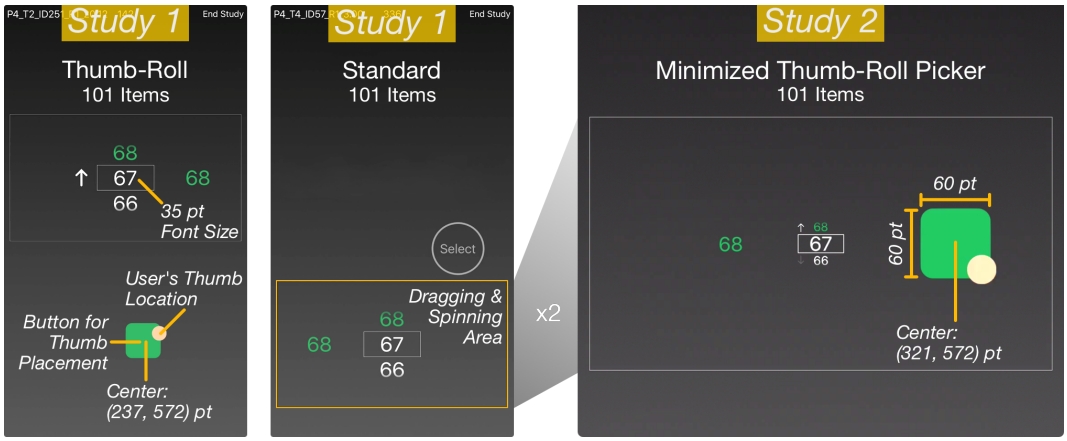


Figure 7.3: UI of our study prototype showing the picker and instructions. Left: Force Picker and System Picker from Study 1. Right: Minimized Thumb-Roll Force Picker, 2× magnified. The iPhone 6s Plus screen measured 414 × 736 pt, or 68 × 122 mm; the origin is located in the top left corner.

cannot be used for confirmation; users had to tap a 'select' button placed within thumb reach above the picker to confirm the selection. Despite continuous scrolling, selection would always snap to the closest discrete value.

For our **Force Picker** (Fig. 7.3, left), we used the same visualization as in the standard picker, but instead of controlling it by dragging and spinning, users were asked to use our force techniques to navigate to the target value shown next to the picker. Users held the device in their right (dominant) hand, and placed their thumb on a green 60 × 60 pt button. When they applied force, the values started scrolling up resp. down. The arrow indicated the current direction. Upon releasing the thumb, the picker snapped to the closest value and selected it, completing each trial.

While participants control the System Picker via dragging and spinning, they control the Force Pickers via pressing on a green button and use *Pulse*, *Press-Through*, and *Thumb-Roll* to change the direction.

For both our standard picker and our Force Picker, we displayed the higher value above, and the lower value below the current item. Although the iOS and Android pickers display values in reversed order, we wanted to achieve a more *natural mapping* [Norman, 2002] for force input: An increase (↑) in force increased (↑) the value more quickly by scrolling more quickly.

For a natural mapping, higher values are displayed above and lower values are shown below the picker center.

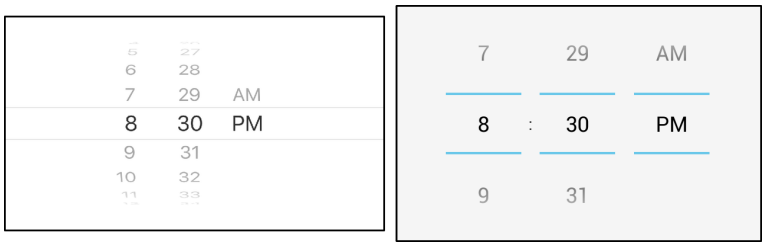


Figure 7.4: Default iOS 10 system picker (left) and Android 7 system picker (right). The iOS picker measures 414 × 216 pt on a 414 × 736 screen (taking up 30% of the screen height) and shows up to seven values at a time, whereas the Android picker measures 620 × 424 px on a 720 × 1,280 px screen (taking up 33% of the screen height) and displays three values per picker wheel at a time.

Transfer Function and Force Pickers

We used the iPhone's built-in force sensor to obtain the intensity of each force touch applied to the touchscreen.

According to Apple's API documentation [Apple Inc., 2019b], the iPhone force sensor API delivers unitless force values between 0 and $\frac{400}{60} \approx 6.67$ in steps of $\frac{1}{60}$, with force sensitivity set to the "medium" default. Values around 1.0 should be interpreted as an ordinary touch; higher values as intentional force input. Although Apple does not state how these values translate to Newtons, experiments [Nelson, 2015] suggest a 4 N range and a linear transfer function. In the remainder of this chapter, we report both the iOS-specific 0–6.67 range and their generic relative values.

For our Force Pickers, we used the same linear transfer function as Wilson et al. [2011], but with a boosting effect upon crossing 80% of the iPhone's force space.

According to the design of our techniques, we used RC for mapping force to the Force Picker scrolling speed. Forces below 1.35, the 20% resting threshold, were ignored to let ordinary touch input coexist and to let users rest their thumb on the touchscreen without affecting value input. We set $Speed(x) = (24.812x - 33.496)\frac{mm}{s}$, $x \in [1.35; 5.34)$, following [Wilson et al., 2011]. For forces beyond 5.34, the 80% boost threshold, we set $Speed(x) = (38.824x + 46.588)\frac{mm}{s}$, $x \in [5.34; 6.67)$, which we determined through pilot tests. Pressing harder than 6.67 resulted in a plateau of 305 $\frac{mm}{s}$ scrolling speed.

We implemented the *Pulse*, *Press-Through*, and *Thumb-Roll* Force Pickers according to our design specifications.

We implemented our force techniques according to our specifications above. To issue a Pulse, users had to start below the resting threshold of 20% of the sensor range, and cross it clearly, reaching 3.34 (50% of the sensor range, determined in pilot tests) at a rate of change of at least 10% between two digitizer frames captured every 16 ms, and then drop quickly below the resting threshold again. Detection for Press-Through was similar, except that users had to reach the maximum measurable force of 6.67 without reducing force quickly afterwards. For Thumb-Roll, we captured the location of the thumb's touch point at resting force. When the touch moved to the right or left, direction was set to up or down, respectively. Since the user's thumb could drift during rolling, we re-calibrated its location whenever force dropped below the resting threshold.

Independent variables were Technique, Range, and Distance.

Independent Variables were Technique (Baseline, Pulse, Press-Through, and Thumb-Roll), Range (10, 30, 60, and 101 items), and Distance (OneStep, 20%, 50%, 80%) in both directions. Range determined the number of available picker values, representing typical use cases: selecting a digit (0–9), a day of the month (1–30), minutes for a timer (1–60), or a percentage (0–100). Distance denoted how many values lay between start and target value. We chose relative Distances to allow comparison across Ranges. Only OneStep was absolute to test single increments and decrements, representing typical off-by-one corrections. For all force techniques, applying force navigated up by default. Figure 7.5 lists all values users had to navigate from and to in both directions. To prevent shortcut strategies by navigating in the opposite direction, we disabled wrap-around. Instead, Distance included OneStep and 20% as short distances. We did not include the extreme values as targets since this would have simplified the task, as scrolling stopped at these values, and since it would not be feasible for pickers that support wrap-around.

Each participant performed 384 trials.

We recorded 4 Techniques × 4 Ranges × 4 Distances × 2 directions × 3 repetitions = 384 trials per participant. We also screen-captured all trials to investigate potential outliers later. Technique and Range were each counterbalanced using a Latin Square, and Distance in both directions was randomized. Once all three repetitions were done, the user continued with the next Range, until all Ranges had been tested. As each Technique was presented to the user, she was allowed to explore it until she felt more familiar with it.

Distance \ Range	10	30	60	101	10	30	60	101
OneStep	6→7	19→20	39→40	67→68	4→3	13→12	26→25	44→43
20 %	1→3	3→9	7→19	11→31	6→4	19→13	39→27	67→47
50 %	3→8	10→25	20→50	34→84	7→2	23→8	46→16	78→28
80 %	0→8	1→25	1→49	0→80	9→1	30→6	60→12	100→20

Figure 7.5: Start and target values per Range × Distance combination.

Dependent variables were the task completion *Time*, the number of *Crossings* when the target value was over- or undershot, and the *Success* of whether the user selected the right value or not. We also captured the thumb *Gesture Footprint* and user's *Ranking* data.

Dependent Variables were *Time* [ms], *Crossings* [$i \in \mathbb{N}$], and *Success* [0,1]. For each trial, *Time* denoted the time from first contact with the touchscreen until value selection by liftoff (force techniques) or tapping the 'select' button (Baseline). *Crossings* counted how often the user overshot the target value (e.g., cf. G. Ramos et al. [2004]), such that she had to change direction and navigate back. *Success* indicated whether the user selected the correct target value (1) or not (0). While we also recorded how far off users were from the target value in those cases, they missed by more than one in only .05% of all trials. We also logged touch positions every 16 ms to capture gesture starting points and *footprints*. At the end, users were asked to *rank* the four techniques by preference (1: most, 4: least).

7.4.1 Results

We identified outliers using the *Tukey Method for Extreme Outliers* and verified them by looking at our screen recordings.

A total of 32 outliers were identified by applying the *Tukey Method for Extreme Outliers* on *Time* (18–63 s). Looking at the screen recordings for these trials revealed that users had their thumb already placed on the touchscreen, which activated time counting, but they were still asking questions before actually performing the task. Hence, these trials were not representative and we therefore excluded them from the analysis. Since we were interested in how performance for each Technique was affected by Range and Distance, we concentrate on these results. Hence, although significant for *Time* and *Crossings* (p <.001, each), we will not discuss main effects for Range and Distance, since these mix data from all Techniques. We log-transformed *Time* for repeated measures ANOVA. Since *Success* was dichotomous, we conducted Cochran's Q and McNemar tests. We analyzed the count for *Crossings* with Friedman and Wilcoxon Signed Rank tests.

Technique had a significant main effect on ***Time*** ($F_{3,6065}$ = 955.73, p <.001). Tukey HSD post hoc pairwise comparisons were all significant (p <.001, each). At 2,547 ms, users were fastest for Baseline. The fastest force technique was Thumb-Roll, yet 900 ms slower than Baseline, followed by Pulse and Press-Through (Fig. 7.6, right).

Using the Force Picker, participants were a significant 900 ms slower than with the System Picker.

There was also a significant Technique × Distance interaction effect ($F_{9,6065}$ = 10.90, p <.001). Tukey HSD post hoc pairwise comparisons revealed that, for each Technique, users were significantly fastest for OneStep, followed by 20%, 50%, and 80% Distance (p <.001, each). This was to be expected, as navigating farther distances takes more time. For each Distance, pairwise post hoc tests showed that users were always fastest for Baseline (Fig. 7.6, left), then Thumb-Roll, Pulse, and Press-Through (Pulse vs. Thumb-Roll with 50% and 80% Distance: both p = .01, all others: p <.001).

As to be expected, navigating farther Distances took more time.

Technique had a significant main effect on ***Crossings*** ($\chi^2(3)$ = 121.41, p <.001). Post hoc pairwise comparisons revealed that users made significantly more *Crossings* for Baseline compared to all force techniques (p <.001, each), but the difference was small (Fig. 7.7, left).

Participants made more *Crossings* with the System Picker compared to the Force Pickers.

There was also a significant Technique × Range interaction effect ($\chi^2(15)$ = 154.27, p <.001). Post hoc tests revealed that users made significantly fewer *Crossings* with Pulse for Range 10 compared to Baseline (p = .001), and they made significantly fewer *Crossings* with Thumb-Roll for Range 30 compared to Baseline (p = .006) (Fig. 7.8, left). Within each Technique, Range did not affect *Crossings*.

Especially for smaller Ranges, participants made fewer *Crossings* with the Force Pickers compared to th System Picker.

There was also a significant Technique × Distance interaction effect ($\chi^2(15)$ = 485.17, p <.001). Post hoc pairwise comparisons revealed that users made significantly fewer *Crossings* for all force techniques compared to Baseline for all percentual Distances (Baseline vs. Thumb-Roll with 50% and 80% Distance: p = .004, Baseline vs. Pulse with 80% Distance: p = .002, all others: p <.001). Only for OneStep, users made significantly more *Crossings* with Press-Through compared to all other techniques (p <.001, each) (Fig. 7.8, right). Also, these tests revealed that for Baseline, Thumb-Roll, and Pulse, users always made significantly fewer *Crossings* for OneStep compared to all other Dis-

For Baseline, Thumb-Roll, and Pulse, participants always made significantly fewer *Crossings* when changing a value by one step compared to all other Distances.

		Range				Distance			
		10	30	60	101	OneStep	20%	50%	80%
Baseline	M	1,435	2,148	3,004	3,606	855	2,151	3,180	4,009
	SD	674	1,204	1,857	2,279	604	961	1,561	1,996
Pulse	M	2,685	3,764	4,691	5,561	1,863	3,899	4,967	5,964
	SD	2,148	2,530	2,950	3,599	1,627	2,714	2,870	3,115
Press-Through	M	4,542	5,699	3,764	8,705	3,967	5,703	7,809	9,163
	SD	4,635	4,391	5,863	6,213	4,665	4,812	5,414	5,876
Thumb-Roll	M	1,983	2,963	4,021	4,863	1,320	2,988	4,343	5,187
	SD	1,185	1,815	2,496	3,101	1,018	1,694	2,514	2,539

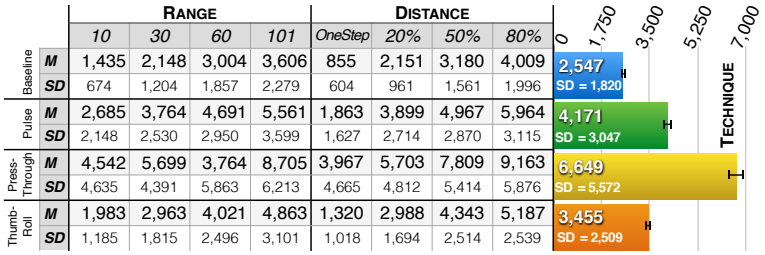


Figure 7.6: *Time* (ms) for Study 1. Left: M and SD for Technique split by Range and Distance. Right: Mean *Time* for each Technique, all significantly different from each other. Error bars denote 95% CI.

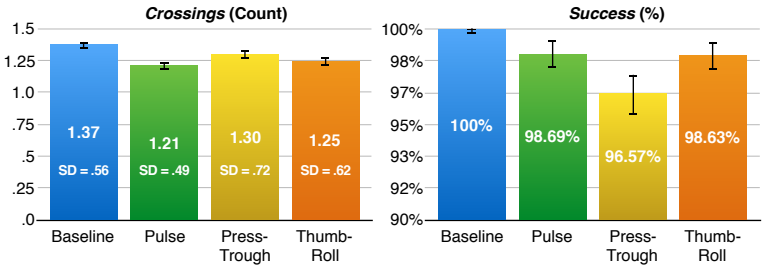


Figure 7.7: *Crossings* (left) and *Success* (right) for Study 1. Users made significantly more *Crossings* for Baseline, but its *Success* was significantly higher than for all force techniques. Error bars denote 95% CI.

tances (Baseline vs. Thumb-Roll with 20% Distance: $p = .012$, Baseline vs. Pulse with 50% and 80% Distance: both $p = .003$, all others: $p < .001$).

While *Success* for the System Picker was 100%, *Success* for *Pulse* and *Thumb-Roll* were close to 99%, hence almost similar.

Technique had a significant main effect on ***Success*** ($Q(3) = 61.69$, $p < .001$). Post hoc pairwise comparisons were not significant between Pulse and Thumb-Roll, but between all other pairs ($p < .001$, each): Users performed with 100% *Success* for Baseline, followed by Pulse (98.7%), Thumb-Roll (98.6%), and Press-Through (96.6%), (Fig. 7.7, right). Although Pulse and Thumb-Roll had significantly lower *Success* compared to Baseline, the difference of less than 1.4% is small.

For each Technique, Range did not effect *Success*.

There was also a significant Technique × Range interaction effect ($Q(15) = 78.40$, $p < .001$). Post hoc pairwise comparisons revealed that users had always significantly higher *Success* with

	Range (M, SD)								Distance (M, SD)							
	10		*30*		*60*		*101*		*OneStep*		*20%*		*50%*		*80%*	
Baseline	1.35	0.55	1.34	0.52	1.36	0.57	1.42	0.60	1.02	0.14	1.54	0.59	1.47	0.60	1.45	0.61
Pulse	1.16	0.44	1.19	0.46	1.22	0.51	1.27	0.54	1.06	0.27	1.28	0.60	1.24	0.47	1.26	0.52
Press-Through	1.34	0.82	1.27	0.70	1.29	0.68	1.30	0.68	1.30	0.72	1.27	0.63	1.32	0.78	1.31	0.75
Thumb-Roll	1.19	0.59	1.21	0.57	1.27	0.62	1.32	0.70	1.07	0.37	1.25	0.58	1.32	0.72	1.34	0.73

Figure 7.8: *Crossings* for Study 1 split by Range and Distance.

Baseline (100%) compared to Press-Through for each Range (10: 97%, $p = .02$; 30: 95%, $p < .001$; 60: 96%, $p < .001$; 101: 98%, $p = .021$). In addition, for Range 30, users had significant lower *Success* for Press-Through (95%) compared to Baseline (100%), Pulse (99%), and Thumb-Roll (98%), ($p < .001$, each). Also, for Range 60, users had significantly higher *Success* for Thumb-Roll (99%) compared to Press-Through (96%), ($p = .035$). Within each Technique, however, Range had no effect on *Success*.

For each Technique, Distance did not effect *Success*.

There was also a significant Technique × Distance interaction effect ($\chi^2(15) = 69.16$, $p < .001$). Post hoc pairwise comparisons revealed that users had always significantly higher *Success* for Baseline (100%) compared to Press-Through for each Distance (OneStep: 97%, $p = .004$; 20% Distance: 96%, $p < .001$; 50% Distance: 96%, $p < .001$; 80% Distance: 97%, $p = .007$). In addition, for 20% Distance, users had significantly lower *Success* for Press-Through (96%) compared to all techniques (Baseline: 100%, $p < .001$; Pulse: 99%, $p = .035$; Thumb-Roll: 99%, $p = .005$) Also, for 50% Distance, users had significantly lower *Success* for Press-Through (96%) compared to Pulse (99%), ($p = .003$). Still, within each Technique, Distance had no effect on *Success*.

Press-Through was significantly least preferred by participants.

Users' ***Ranking*** ($\chi^2(3) = 29.25$, $p < .001$) revealed no significant differences between Baseline ($M = 1.56$, $SD = .90$), Thumb-Roll ($M = 1.88$, $SD = .89$), and Pulse ($M = 2.06$, $SD = .68$) ($p < .001$, each). Press-Through ($M = 3.63$, $SD = .50$), however, was significantly least preferred over Baseline ($p < .001$), Pulse ($p = .003$), and Thumb-Roll ($p < .001$).

The *Pulse* Force Picker caused the smallest *Gesture Footprint*.

Figure 7.9 visualizes the ***Gesture Footprint*** from all users for each Technique. For our force techniques, the green square marks where users were asked to place their thumb. Red data points represent first contact with the screen (*Touch Began*), blue data points all other touches (*Touch Moved*). As can be seen, the footprint for the force techniques is much smaller than for

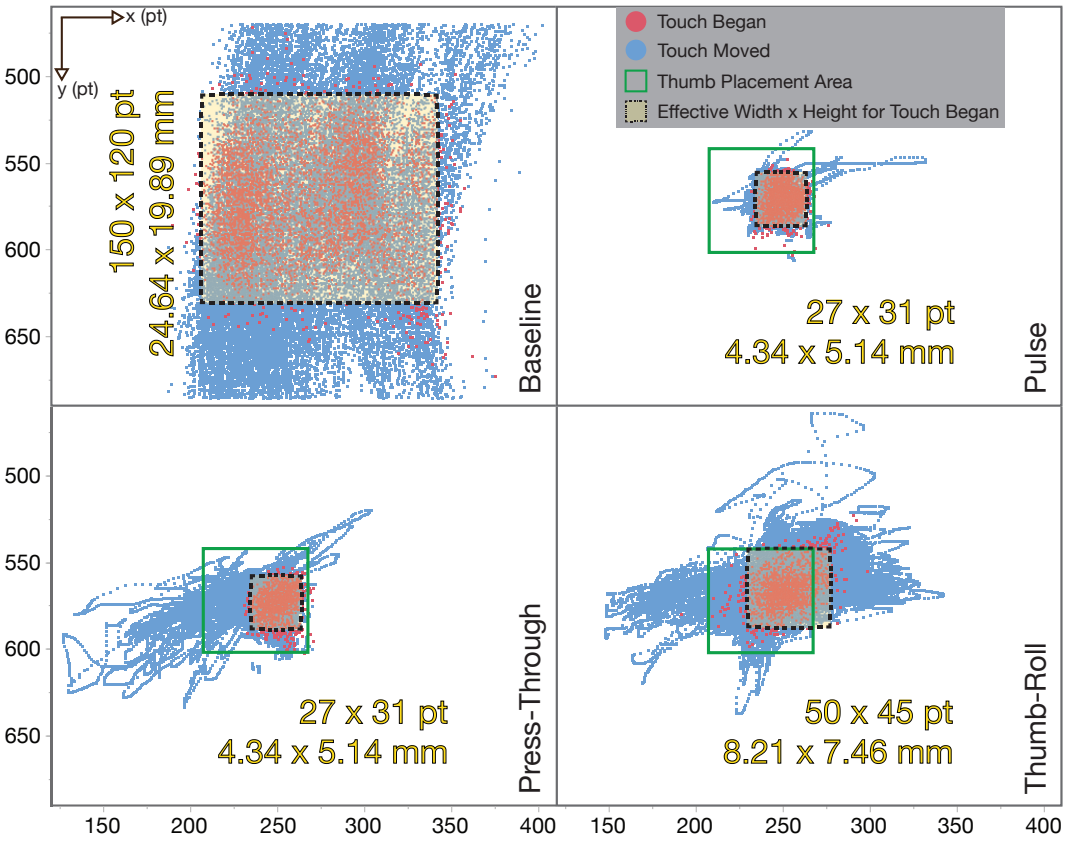


Figure 7.9: Gesture Footprint from Study 1. Our three Force Picker techniques (right) have much smaller footprints than standard dragging and spinning (left). Yellow overlays denote 96% of all initial touch points. Compared to Baseline, the force techniques require only 5–12% of the space.

Baseline. Based on all first touches, we also calculated the effective width × height (yellow overlays in Fig. 7.9) capturing 96% of these data points, multiplying the SD in x and y by 4.133 [Soukoreff et al., 2004]. These rectangles denote the space that needs to be reserved for detecting the start of an interaction with the Force Picker. During interaction, users may drift (blue dots), and touch input outside the rectangles should not be passed on to other widgets until thumb liftoff ends this modal interaction. Note that content beneath the blue dots will be occluded by the thumb. With the standard picker, users do multiple liftoffs while spinning and dragging, whereas with our force techniques, each selection requires only one liftoff. Thumb-Roll required only 12%, Pulse and Press-Through only 5% of the effective area (width × height) required for Baseline.

7.4.2 Discussion

For our force techniques, users performed fastest (3.5 s) and most accurate (99%) with Thumb-Roll. Although users' *Success* for Pulse was the same, they were a significant 700 ms slower. For Baseline, users were 900 ms faster than for Thumb-Roll, and achieved 100% *Success*. However, this was to be anticipated, since all users were familiar with standard picker interaction, but not with force control, and post-corrections after liftoff are part of the interaction with standard pickers. Users' slower task completion time for our force techniques could also be explained by *Crossings*. Users reported that overshoots felt easier to correct with dragging and spinning than by applying force, and therefore navigated more carefully with the force techniques. For single value increase or decrease, however, users could gently tap using our force techniques, as this would cause the picker to scroll a little but then snap to the next value. This could explain why users made significantly fewer *Crossings* compared to further Distances. For Press-Through, however, tapping for a one-step change was not possible when decreasing values. This technique was also problematic for users when decreasing values because of sensor limitations: If they had applied force beyond what the sensor could detect, as they were beginning to release, the system could not provide continuous feedback, although this is essential for force input [Wilson et al., 2011]. Users then tried to reduce force more quickly, which resulted in overshooting, hence an increase in *Crossings* and *Time* needed for corrections. Furthermore, users were rather confused that the force-to-value mappings reversed depending on the direction, and did not consider the natural mapping between force increase/decrease and value increase/decrease a benefit. Consequently, users ranked Press-Though lowest among all techniques.

The *Press-Through* Force Picker was problematic for participants because upon direction change, the force sensor would be driven into saturation such that no visual updates could be communicated to the participant until the force dropped below the maximum of the force range.

As expected, the gesture footprint for all force techniques was drastically smaller than for dragging and spinning. Naturally, Thumb-Roll had the largest footprint among force techniques, since rolling requires more space than stationary Pulse or Press-Through gestures. Nevertheless, Thumb-Roll accounted for only 12% of the effective width × height by Baseline. In all, since Thumb-Roll achieved the best *Time*, *Success*, and user *Ranking* of our force techniques at a fairly small gesture footprint, we further pursue this technique in the rest of this chapter. If the

The *Gesture Footprint* for all Force Pickers was drastically smaller compared to the footprint caused by spinning and dragging the System Picker.

footprint needs to be even smaller, Pulse is an alternative to consider, but at the cost of *Time*.

Now we want to reduce the display footprint of our best Force Picker, the *Thumb-Roll* picker.

The promising reduction in *gesture* footprint from Thumb-Roll led us to investigate whether we could now effectively save screen space by also reducing the *display* footprint of the Force Picker. Therefore, we next compared user performance for Thumb-Roll in our standard-sized Force Picker to using it in a minimized Force Picker.

7.5 Study 2: Minimizing the Force Picker

In Study 2, we reduced the size of the *Thumb-Roll* Force Picker from Study 1 and adjusted the transfer function accordingly.

We modified Study 1, using only Thumb-Roll as technique and adding Size as independent variable to represent the 414 × 216 standard-sized picker and a 44 pt squared minimized Force Picker (7.3 × 7.3 mm) that fits within the height of an iOS table row (Fig. 7.3). We adjusted the transfer function for the minimized Force Picker to $Speed(x) = 8.271x - 11.165\frac{mm}{s}$, $x \in [1.35; 5.34)$ (normal) and $Speed(x) = 12.941x + 15.529\frac{mm}{s}$, $x \in [5.34; 6.67)$ (boosted) to achieve the same scrolling for both pickers. Based on the gesture footprint from Study 1, the thumb placement area was shifted a little more to the right (Fig. 7.3, right), such that the thumb would not occlude the picker during interaction. Levels for Range and Distance were identical to Study 1. Again, *Time*, *Crossings*, *Success*, and *Gesture Footprint* were recorded. We also asked users whether they found the minimized Force Picker hard to read (7-point Likert scale, 7 = totally agree).

Each of our eight participants performed 192 trials in which they selected values using the standard-sized vs. the minimized *Thumb-Roll* Picker.

Of our eight participants (24–40 years, $M = 31.13$, $SD = 6.51$, all right-handed, three females, none from Study 1), four started with the minimized Force Picker. Counterbalancing, randomization, and presentation order of conditions were identical to Study 1. We recorded 2 Sizes × 4 Ranges × 4 Distances × 2 directions × 3 repetitions = 192 trials per user, excluding two test trials for each Size. Users also had the opportunity to briefly familiarize themselves with both pickers beforehand.

7.5.1 Results

Data analysis was performed similar to Study 1, directly contrasting effects for each Size. We again log-transformed *Time* for a repeated-measures ANOVA.

We performed the same data analysis as Study 1.

Size had a significant main effect on ***Time*** ($F_{1,1514}$ = 9.86, p = .002): Users needed M = 2,698 ms to complete a trial for the standard-sized Force Picker, and M = 2,988 ms for the minimized version (Fig. 7.10, left). There were no interaction effects on *Time*.

Participants were about 290 ms faster with the standard-sized *Thumb-Roll* Picker.

There was a significant Technique × Range interaction effect on ***Crossings*** ($\chi^2(7)$ = 30.21, p <.001), but post hoc tests were not significant. There was also a significant Technique × Distance interaction effect ($\chi^2(7)$ = 61.92, p <.001). Post hoc tests showed that users made significant fewer *Crossings* for OneStep (M = 1.06, SD = .67) compared to 50% Distance (M = 1.34, SD = .27) for the standard-sized, and for the minimized Force Picker, users made significant fewer *Crossings* for OneStep (M = 1.08, SD = .35) compared to 80% Distance (M = 1.50, SD = .97) (both p <.001).

As to be expected, participants made significantly fewer *Crossings* for single value increase or decrease compared to navigating larger Distances.

There were neither significant main effects nor interaction effects for ***Success***. Users correctly selected values with a *Success* of 96.6% for the standard-sized and 96.1% for the minimized Force Picker (Fig. 7.10, right).

The size of the picker had no effect on *Success*.

The effective width × height based on the ***Gesture Footprint*** from first touch contact measured 43 × 53 pt for the standard-sized Force Picker (Fig. 7.11, left), which was slightly larger than the 43 × 42 pt for the minimized version (Fig. 7.11, right). These results are similar to Study 1.

The *Gesture Footprint* for the minimized *Thumb-Roll* picker was slightly smaller than for the standard-sized counterpart.

Overall, users disagreed that the minimized Force Picker was hard to read (M = 3.38, SD = 1.51); only one user explicitly stated reading difficulties.

One participant found the minimized picker hard to read.

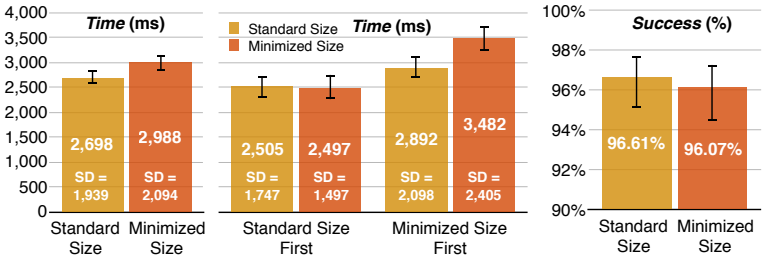


Figure 7.10: *Time* and *Success* for Study 2. Users were 290 ms faster with the standard-sized Force Picker (left), but starting with the standard size also sped up using the minimized version afterwards (center). *Success* was the same for both Sizes (right). Error bars denote 95% CI.

7.5.2 Discussion

Study 2 participants were faster participants from Study 1.

Users were 290 ms slower with the minimized Force Picker than with the standard-sized Force Picker. Nevertheless, for both Sizes, users from Study 2 were 470–750 ms faster compared to Thumb-Roll performance from Study 1.

Testing the standard-sized picker first had a positive training effect on then using the minimized picker.

We wondered whether this was due to a training effect, since users had to perform twice as many Thumb-Roll trials in Study 2. Therefore, we split the *Time* data: Users who tested the standard-sized Force Picker first performed trials equally fast for both Sizes (Fig. 7.10, center): 2,505 ms for the standard-sized Force Picker and 2,497 ms for the minimized Force Picker—which was actually as fast as dragging and spinning in Study 1. However, at 3,482 ms, users testing the minimized Force Picker first were about 1 s slower than the other group, but improved their performance while testing the standard-sized Force Picker afterwards (M = 2,892 ms). It seems that users testing the minimized Force Picker first navigated more carefully because they were confronted with two new situations at once, Thumb-Roll force input and the small picker visualization, while those testing the standard-sized Force Picker first encountered them one by one, increasing confidence.

Success was slightly worse than findings from Study 1.

Crossings and *Success* were similar for both Sizes, although *Success* was slightly worse (96%) than with Thumb-Roll in Study 1, possibly due to a time–accuracy trade-off.

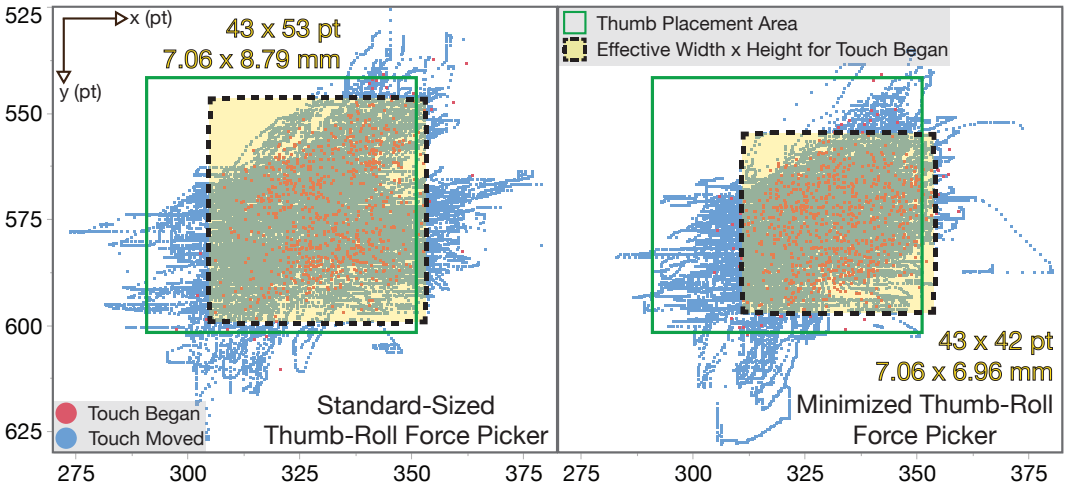


Figure 7.11: Gesture Footprint for Study 2. Left: Footprint for the standard-sized Thumb-Roll Force Picker. Right: Footprint for minimized Thumb-Roll Force Picker. The yellow overlays denote 96% of all first touch contact points and were similar for both pickers.

Based on these findings, we wondered whether users could become faster with further training, since users from both studies so far had known neither Thumb-Roll force input nor the minimized Force Picker visualization before. Therefore, we conducted a third study with trained users comparing the minimized Thumb-Roll Force Picker against a standard picker.

Next, we wanted to understand whether further training can help users become faster in controlling the minimized *Thumb-Roll* picker.

7.6 Study 3: Trained User Performance

We recruited four participants (29–35 years, $M = 32.50$, $SD = 2.65$, all right-handed, one female, none from previous studies) to complete ten training sessions at home over five days to master Thumb-Roll for the minimized Force Picker. We modified Study 2 to capture Distances from 10–90% (in 10% steps) as well as OneStep, again for both directions. We picked the same Range sizes. Users performed 4 Ranges × 10 Distances × 2 directions × 10 sessions = 800 trials. On average, the training took 1:10 h. Afterwards, users did a final session at our lab, using the tasks from studies 1 and 2, but testing the Baseline standard picker from Study 1 against the minimized Thumb-Roll Force Picker from Study 2.

We let four participants train value selection with the minimized *Thumb-Roll* Force Picker for more than one hours across five days.

7.6.1 Results

Time decreased with training experience.

Figure 7.12 shows how users' *Time* decreased over the ten training sessions, while *Success* remained at about 98%. For the analysis, we will focus on the results from the final test, again by directly contrasting effects for each Technique. *Time* was again log-transformed for repeated-measures ANOVA.

After training, users were faster with the minimized *Thumb-Roll* picker than with the System Picker.

Technique had a significant main effect on ***Time*** ($F_{1,751} = 143.20$, $p < .001$): Users needed $M = 2{,}300$ ms to complete a trial for Baseline, but only $M = 1{,}796$ ms for Thumb-Roll (Fig. 7.13, left).

Results for *Crossings* were similar to findings from Study 1.

Technique also had a significant main effect on ***Crossings*** ($Z = 3062.50$, $p < .001$). As seen in Study 1, users made significantly fewer *Crossings* using Thumb-Roll ($M = 1.16$) than with Baseline ($M = 1.35$), but the difference was small (Fig. 7.13, center). There was also a significant Technique × Range interaction effect ($\chi^2(7) = 35.27$, $p < .001$), but post hoc tests were not significant. There was also a significant Technique × Distance interaction effect ($\chi^2(7) = 107.98$, $p < .001$). Post hoc tests revealed that within Technique, users made significantly fewer *Crossings* for OneStep (each Technique: $M = 1.01$, $SD = 1.02$) compared to all other Distances (Baseline: $M = 1.44–1.50$, $SD = .54–.63$, Thumb-Roll: $M = 1.19–1.24$, $SD = .40–.43$), ($p < .001$, each).

Participants achieved the same *Success* for both pickers.

There were neither significant main effects nor interaction effects for ***Success***: Although Baseline had again 100% *Success*, this was not significantly different from the 99% for Thumb-Roll (four errors were made in total).

The *Gesture Footprint* for the minimized *Thumb-Roll* picker is only 6% of that of the System Picker.

Figure 7.14 shows ***Gesture Footprint*** and effective width × height based on first touch contacts for both Techniques. The minimized Thumb-Roll Force Picker takes up only 40 × 25 pt, 6% of the 131 × 134 pt dragging and spinning requires. While the low number of users leads to lower variance, Figure 7.14 shows a very consistent gesture footprint among these trained users.

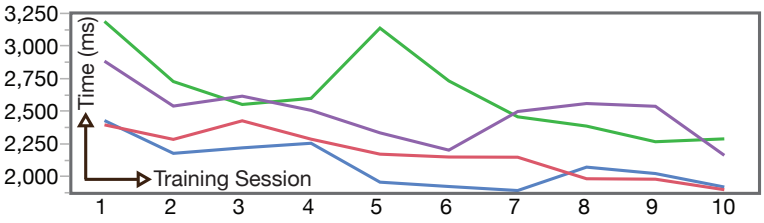


Figure 7.12: *Time* (ms) for each training session from four users before performing a final test in Study 3. Users became faster over ten sessions.

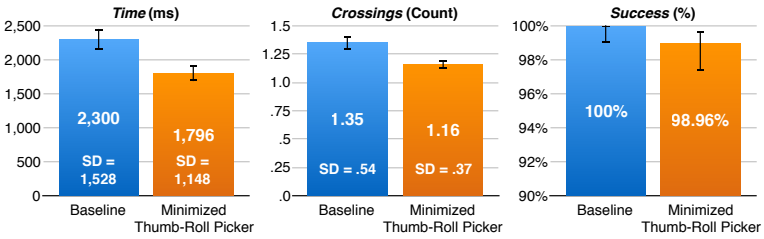


Figure 7.13: *Time* (ms), *Crossings*, and *Success* (%) for Study 3. Overall, users were faster and made fewer *Crossings* with Thumb-Roll compared to Baseline, while *Success* was not significantly different between both TECHNIQUES. Error bars denote 95% CI.

7.6.2 Discussion

Using force input, values can be navigated very quickly when using a "pumping" strategy.

The results indeed show that, with some training, users can become faster with the minimized Thumb-Roll Force Picker than with the standard picker. Task completion time for dragging and spinning the standard picker was the same as in Study 1, but it was now outperformed by our minimized Thumb-Roll Force Picker, for which users were 500 ms faster. All users reported that they developed a "pumping" strategy to get faster: Instead of applying a strong force over a long time, they pressed hard, then released a bit, and then pressed harder again, to navigate far distances. While not significant, the differences in *Time* between both TECHNIQUES increased with RANGE, from 177 ms for RANGE 10 to 240 ms for RANGE 101, and with DISTANCE, from 0 ms for OneStep to 240 ms for 80% DISTANCE. Hence, Thumb-Roll paid off especially when navigating larger RANGES or DISTANCES, apparently due to the speed boost that applying strong

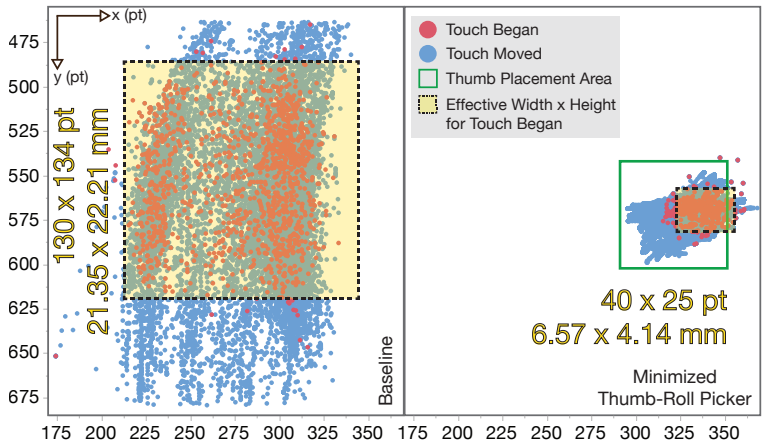


Figure 7.14: Thumb Gesture Footprint for Study 3 for the Baseline picker (left) and the minimized Thumb-Roll picker (right). The footprint for controlling the minimized Thumb-Roll Force Picker was much smaller than dragging and spinning. The yellow overlays denote 96% of all first touch contact points: Thumb-Roll required only 6% of the space that users needed for dragging ans spinning the Baseline picker.

force triggered. *Success* was not significantly different between both Techniques ($\chi^2(1) = 2.25$), and was again similar to Study 1.

Training helped participants to become experienced in using the minimized Thumb Roll Force Picker, such that they were even faster than using the System Picker.

In summary, our studies show that Thumb-Roll was the most promising technique for a force-based picker on force-sensitive smartphones, while requiring only a fraction of the gesture footprint (6%) and display footprint (20% of the height) of a standard picker as used on these systems today. Taking *all* touch points from thumb-rolling into account, trained users will cover an area of 10.2 × 5.3 mm, such that the Force Picker should be placed farther than this to avoid occlusion by the thumb. After 1:10 h of training, users were able to select values faster with the minimized Thumb-Roll Force Picker than with the standard picker without significant loss of accuracy (*Success* rates of 99% for the minimized Thumb-Roll Force Picker vs. 100% for the standard picker).

Below, we present some application examples that could benefit from using the minimized Thumb-Roll Force Picker instead of the system picker.

7.7 Application Examples

Space-Efficient In-Row Value Selection

As illustrated in Figure 7.1, our Force Picker can be used to pick values without moving context off-screen like standard pickers do: To add a new calendar entry, for example, the user needs to specify a date, start time, and duration. With standard pickers, the user taps on, e.g., the start time, and the associated picker slides in, moving contextual information like the end time or invitees down and often off-screen to make space for the picker.

Standard pickers, e.g., for setting a date and time for a calendar entry, require significant screen space.

Using our Force Picker, however, we can keep all that information in place. When the user places her thumb on the hour digits of the start time, the Force Picker appears in the white space next to the label in the same row. It shows the hour currently selected, with the next values above and below it. It needs no more vertical space than the value label it is controlling, so nothing else on the screen moves around, which keeps important contextual information on screen and makes the interaction more serene. Using Thumb-Roll, the user now adjusts the hour through the picker, starting at the value the label displayed. Lifting the thumb fades out the Force Picker again, and updates the label to show the selected value. Minutes or dates can be selected similarly. Any mobile apps that work with times and dates, whether for searching flights and hotels, retrieving credit card statements, using public transport, or planning a drive, can benefit in a similar way.

Our Force Picker requires only little gesture and display space, such that no contextual information needs to be pushed off screen.

Parameterized Shortcuts

Apple's recent smartphones let users apply a "force touch" on certain icons to display a list of context-specific shortcuts. However, since Apple only distinguishes two force levels in its UI, choosing from that list then requires either dragging or a secondary tap. The list also covers any underlying content that may have been useful for context, and the length of the list is limited by the size of the screen to avoid scrolling. An example is setting a countdown timer via force-touching the timer icon in the iOS Control Center: the user can only pick a timer for one, five, 20, or 60 minutes from the list that appears.

Using the Force Picker, one can set a countdown timer straight away by pressing on the according app icon on the device Home Screen.

To set the timer, the user presses to scroll the minutes and rolls the thumb to correct for an overshoot. Lifting the thumb immediately starts the timer.

Using the Force Picker instead, force-touching on the countdown timer icon can fade in the Force Picker above the thumb, letting the user pick a value between 1 and 60 minutes directly by using force and the Thumb-Roll technique. Once she reaches the desired countdown time, she lifts her thumb off the screen to start the timer. Similarly, our Force Picker can be used in home control apps, e.g., to quickly set the volume of a stereo between 0 and 100%.

Self-Paced Browsing

The Force Picker can reduce widget display space to make space for immersive applications like a photo browser app.

Immersive applications like photo browsers or video editing apps should use the screen real estate for content, and reduce the footprint of widgets and other "debris" to a minimum. Therefore, browsing through a set of photos usually requires swiping left or right on the screen repeatedly. However, this still occludes the content and can be tedious and slow.

The Force Picker could also be used to navigate videos at different granularities, e.g., from quick jumps to frame-by-frame navigation.

Using the minimized Thumb-Roll Force Picker instead, the user can browse photos at her desired pace to find the picture she is looking for, while occluding only the smallest possible area (a touch point) of the picture. Similarly, the technique can be used to flip through pages in an ebook or PDF. Indeed, this may remind the user of "thumbing" through pages in a physical book, because both the rolling thumb movement and the application of measured force are somewhat similar. Trimming a video clip can be simplified in the same way: When placing the thumb in the lower left corner of the screen, the Force Picker lets the user find the frame to set the first cut mark using Thumb-Roll. She selects it by lifting her thumb off the screen. The same gesture in the lower right corner of the screen defines the end frame of the video clip.

While these examples assume displaying the currently selected photo, page, or frame in real time, the Force Picker can also display their numbers above the user's thumb.

7.8 Limitations and Future Work

We only investigated the selection of values when they appear in an anticipated order.

In our studies, we only tested input for ordered values, and using only the picker as the form of visualization. However, we expect similar results for related visualizations, such as tables, and also for picking alphabetically ordered strings, like selecting a country name from a drop-down list when entering a shipping address on a website, as long as the user can approximately anticipate how far she has to navigate from the current value to the target value.

Unlike most standard pickers, our picker implementations ordered values bottom-up.

Unlike most standard pickers, our picker visualizations ordered values bottom-up. We used this reversed order to achieve a more natural mapping for force input. Somewhat surprisingly, this turned out not to be an issue, although all participants were familiar with standard pickers. Six users launched discussion after the study, and when asking them whether they noticed the reversed mapping, they all denied.

Due to its size, our minimized Force Picker can only show the previous, the current, and the next value at a time.

We only displayed three values at a time, like the Android system picker, to allow us to minimize the picker for studies 2 and 3, which would not have worked with the seven items the iOS 10 system picker shows. The physical size of our standard picker, and the spinning gestures, were using the iOS system defaults. Interestingly, Apple also seems to have noticed that their system picker required a significant amount of screen space, since for iOS 11, its height has been reduced to 126 pt (20.89 mm). This matches the effective height that we derived from studies 1 and 3 for our standard picker. Hence, the iOS 11 picker can only show five items at a time for each picker wheel, with the highest and the lowest value vertically compressed.

By design, confirming input with a standard picker requires tapping a button.

For our standard picker, users had to tap on a 'select' button to confirm value input, but existing pickers also require the user to tap somewhere outside the picker to confirm the value and close them. Tapping accounted for 275 ms (SD = 40 ms) of the trial completion time. Note, however, that our force techniques also included trial selection time. According to our findings presented in Chapter 5, confirming force input by lifting the thumb requires 240 ms.

The iPhone SDK provides no true touch ellipsis data.

The iPhone touch sensor only reports the center of each touch point, not the entire ellipsis. Actual footprints are therefore slightly larger, depending on thumb size.

The Thumb-Roll Force Picker must be placed away from screen edges.

Using Thumb-Roll directly at the edge of the smartphone screen will not work when the thumb moves beyond the digitizer sensing area. Therefore, the input area for the Thumb-Roll gesture should be placed accordingly.

Since force control becomes more difficult with age, We want to investigate how the performance of elderly participants compares to the results from our studies.

In future work, we would like to reevaluate Press-Through with a stronger force sensor that the user cannot drive into saturation. This way, we could provide immediate feedback when reducing force from any attainable force level. Finally, we would also like to examine whether our results also scale to elderly people: Force input is more difficult for this group to control [Kinoshita et al., 1996], and the minimized Thumb-Roll Picker might turn out to be too small to read.

7.9 Conclusion

We presented an efficient approach for selecting values on force-sensitive handheld devices that saves both gesture and display space.

We presented a novel way to reduce the screen real estate for picking values from long, ordered ranges on smartphones. While existing controls, like pickers or tables, require a significant amount of gesture footprint for dragging and spinning, we exploited the force sensors on modern smartphones to reduce the gesture footprint to the size of the thumb. We conducted three experiments to find a force-based technique that allows for bidirectional control of such a picker, while still fitting inside a standard table row. Our first experiment identified Thumb-Roll, a technique that changes direction upon gently rolling the thumb to the left or right before applying force, as the fastest and most accurate technique to control a standard-sized picker by force input. Although users were slower, accuracy was almost identical, and Thumb-Roll reduced the gesture footprint by 88%. We then showed that this allows the Force Picker to be shrunk down to a minimum size, without affecting accuracy, although it slightly slowed down untrained subjects. Our final study showed that trained participants can actually be significantly faster using the minimized Thumb-Roll Force Picker than with a standard spin & drag picker, without significant loss of accuracy. With Thumb-Roll, these users needed only 6% of the gesture space required

for dragging and spinning, and the minimized Force Picker takes up only 20% of the height of the iOS 10 system picker. We provided application examples for value input on smartphones that benefit from the much smaller footprint and in-place touch interaction of the minimized Thumb-Roll Force Picker. We hope that our findings inspire other researchers and practitioners to further improve our key daily interactions with our smartphones through their force-sensing capabilities.

We have introduced new interaction techniques for handheld touchscreens that utilize the benefit of force input that brings higher expressiveness with each touch point compared to classic touch input. Our techniques enable multi-finger use, solve reachability issues on modern smartphones, optimize input and output space on handheld touchscreens, and they can make the interaction more efficient in many aspects. In the next chapter, we summarize the thesis and draw a conclusion. Furthermore we look at potential future interactions with handheld devices via force input.

In the next chapter, we summarize the thesis and look at future perspectives of force-based interaction on handheld devices.

8

Summary, Perspectives, and Closing Remarks

Interaction with handheld devices is predominantly steered by touch input with our fingers on a touchscreen. While using our hands and fingers is the most natural way of human-object interaction [Kivell, 2015], on handheld devices it is typically constrained to flat touch gestures, such as tapping, swiping, or dragging. This is because the touchscreen is a flat glass surface and it basically only senses *whether* a finger is touching the surface and *where* it touches the surface. However, touch has further properties that could be used for interaction. One of them is *force*, i.e., the intensity with which the user touches the touchscreen surface. Incorporating the user's touch force into the interaction allows her to become more expressive: With each touch, the user not only specifies the location on the screen but also a force value, depending on how hard she is pressing against the surface. Hence, at a single touch location the user can express different inputs just by varying her finger force.

Touch interaction with handheld devices neglects a powerful property that makes touch input more expressive: the intensity of a touch, i.e., its *force*.

8.1 Summary and Conclusion

In this thesis, we set out to exploit the force property and design a set of interaction techniques that make users' touch interaction on handheld devices more expressive and efficient. Next to giving a deep background about force input and an exten-

We designed force-based interaction techniques for one- and two-handed use.

sive overview of existing force-based interaction techniques, we designed and investigated techniques for both one- and two-handed device use that exploit continuous force sensing on the entire front or back of the device (BoD).

BackXPress is an interaction technique for two-handed smartphone use that enables users to utilize their fingers that otherwise rest at the back of the device, for input.

We first presented *BackXPress*, an interaction technique designed for the two-handed use of handheld devices held in landscape orientation. When interacting in landscape mode, e.g., for typing and gaming, users can only use their two thumbs for input at the front, while all other fingers remain at the BoD , holding it in place. BackXPress lets the user utilize these resting fingers for input to augment the interaction with the frontal touchscreen. To design and investigate our technique, we conducted a series of three user studies: The first study aimed at finding out which fingers are suitable for BoD force input and found the index, middle, and ring finger from both hands to work best. The second study revealed where exactly users place and press these fingers at the BoD. The third study investigated users' performance with BackXPress. We found that users can control up to three levels of force per finger at the BoD with 87% accuracy while confirming the selected level via tapping at the frontal touchscreen. Using a correction model that averages the continuously sampled force readings 500 ms before tapping at the touchscreen increases the selection accuracy to more than 91%.

An apt use case for BackXPress is the quick insertion of emojis while typing text.

We presented various use cases for BackXPress that make users' interaction with handheld devices more efficient. One apt example is inserting an emoji while typing text. Without BackXPress, the user must typically operate a menu to change the keyboard layout from text input to emoji characters. This puts the user in a permanent mode that she has to exit by re-selecting the text keyboard layout after the emoji has been inserted. This process is not only susceptible to mode errors for the user, but it is also tedious and consumes time. With BackXPress, the user just presses with a finger at the BoD to pop over the emoji keyboard. To insert an emoji, she simply tap on it with thumb at the front and then releases BoD force to automatically flip back to the text keyboard layout.

Quick Release (QR) is an established and time-efficient mechanism to confirm

While with BackXPress, users confirm their force input at the BoD via touch input at the front, this mechanism is unfeasible when the handheld device has force sensing integrated into the frontal touchscreen and the user is holding the device with a sin-

gle hand. In this case, only the thumb is available for both force control and confirmation. While there exist various single-finger confirmation mechanisms for force input, the most convenient is *Quick Release (QR)*: to confirm the finger's current applied force, the user quickly lifts her finger off the force sensor.

force input using a single finger.

However, QR often does not correctly confirm the user's applied force. Error rates up to 40% are reported in the literature. However, the literature does *not* report how exactly to implement a reliable QR mechanism. Therefore, we set out to challenge these two problems. We collected continuously logged force timestamps from users performing the QR gesture for different levels of force that they were asked to apply. Based on this data we were able to identify the exact duration of the QR gesture and could then infer the force that the user was executing right before lifting the finger off the force sensor. In a verification experiment, we confirmed an accuracy rate of 97% for our data-driven QR implementation.

We challenged the low reliability of QR mentioned in the literature and present a QR implementation with a 97% success rate.

As a next step, we used this confirmation technique as part of our interaction technique *ForceRay (FR)* that addresses a recurring problem with single-handed smartphone interaction: Over the years, smartphones have kept growing in size, with current models featuring touchscreens as large as 6.5 inches. While this large screen real estate is a blessing for output, it is a curse for touch input since the user cannot comfortably reach everywhere with her thumb without re-grasping the device. FR tackles this issue by exploiting the increased expressiveness of force input: When the user slightly presses with her thumb against the force-sensitive touchscreen, she can activate a virtual thumb extension, i.e., a ray, that is casted from the lower screen corner through the touch point to the opposing screen edge. Using this ray, the user can select any of the underlying targets without further finger movement. With the ray comes a cursor that the user controls via force. The harder the user presses, the further the cursor moves along the ray towards the opposing screen edge. Whenever the cursor exits an underlying target, the next target is automatically preselected. To confirm the preselected target, the user simply performs the QR gesture.

We presented *ForceRay (FR)*, a force-controlled virtual thumb extension that enables one-handed touchscreen use without the need for changing the device grip. It uses our QR implementation to quickly confirm selection of distant on-screen target.

In two user studies, we tested users' selection time and accuracy for selecting out-of-reach targets with FR and other reachability techniques that are not based on force input. While FR

In particular, FR is very efficient for selecting targets located at

screen edges and corners, such as menus and navigation buttons. The user can select all targets with ergonomic-pleasant thumb control while maintaining a steady device grip.

was not the fastest technique, we could show that with decent practice users can master the technique and become faster in selecting targets. In particular, FR is efficient for selecting targets located at screen edges and corners, such as menus or navigation buttons. Overall, FR enabled users to reach all targets with ergonomic comfort without the need for re-grasping the device. Furthermore, FR helped users maintaining a steady device grip, which is beneficial for interacting with augmented reality apps or other camera-based applications, since the user can keep the viewport static throughout the interaction.

Selecting values from a large range via touch input requires significant display and gesture space on the touchscreen, such that contextual information must often be temporarily pushed off the screen.

FR uses an absolute force-to-value mapping for controlling the cursor and selecting one of a few targets that are crossed by the ray. However, there are also situations in which the user needs to select one out of many targets, such as when setting a date, time, or percentage via the touchscreen. Typical controls for such value selection are pickers that the user spins by dragging gestures to iterate through the values. However, performing such dragging and spinning requires a significant amount of gesture and display space that the pickers occupy on the touchscreen. To optimize that space, the pickers are often shown upon demand, which causes contextual information to be temporarily pushed off the screen.

Our solution, the *Force Picker* not only requires just a fraction of space needed by standard pickers. It also makes value selection faster.

We addressed this issue with our *Force Picker* that lets the user select one out of many values via force input. This way, value selection only requires the gesture space of a single touch point. Since selecting a value from a large range is not feasible with an absolute force mapping, we used a relative force-to-velocity mapping that maps force to the speed of spinning through the value range. We evaluated our Force Picker in three user studies.

This is due rate-based control, for which we designed the *Thumb-Roll* mechanism that lets users determine the direction of iterating through the value range by rolling their thumb prior to applying force.

The first study investigated three mechanisms for changing the direction of force-controlled spinning, since a relative mapping by default only allows for unidirectional control. Participants preferred our *Thumb-Roll* mechanism that sets the direction by rolling the thumb to the right side to increase the values and by rolling to the left side to decrease the values when the applying force. While this Thumb-Roll Force Picker significantly reduced the required gesture space compared to a standard System Picker, we then also reduced the display space by introducing a minimized Thumb-Roll Force Picker. Yet, compared to the standard-sized counterpart, participants were a 300 ms slower in

selecting values. To let participants familiarize with force control and the small display size of the picker, we let them intensively train the minimized Thumb-Roll Force Picker against using the standard System Picker. After one hour of training, participants were faster in selecting values with the minimized Thumb-Roll Force Picker than with the familiar System Picker. Furthermore, the minimized Force Picker requires only 6% of the standard System Picker gesture space and 20% of its display space.

8.1.1 Contributions and Benefits

We designed four techniques that make handheld touch input more expressive and efficient.

In conclusion, we designed and investigated four different techniques that make touch input on handheld devices more expressive by incorporating the intensity of touch against the touchscreen as an adjustable parameter. Our techniques benefit handheld users in different ways for day-to-day interaction:

- *BackXPress* empowers users to incorporate otherwise unused fingers into the interaction, making them efficient in quickly accessing menus and functions without explicit activation and deactivation of modes **(H1, H4)**.
- Our documentation and implementation of the *Quick Release* mechanism makes confirming force input efficient without compromising accuracy **(H4)**.
- With *ForceRay*, one-handed handheld touchscreen use becomes ergonomic-pleasant while accessing targets at edges and corners of the screen can be acquired quickly despite their distance to the user's finger **(H2)**.
- Our minimized Force Picker makes value selection not only faster, but also makes efficient use of display space on handheld touchscreens, leaving space for showing other relevant content **(H3, H4)**.

Force input requires decent practice.

However, as we have learned from our studies, force input requires decent practice before one can fully control it. Only then the aforementioned benefits actually pay off.

8.2 Future Perspectives

Current handheld device interaction equals touching and pressing against a rigid, flat glass surface.

Looking at the future of force-based interaction on handheld devices implies also looking at what handheld devices might look like anytime soon. Currently, interaction on handheld devices means touching or pressing against a flat, rigid glass surface. This takes a lot from the natural sensation that we usually experience when interacting with our hands and fingers with physical objects [Kivell, 2015]. Objects are made of different materials and provide a variety of qualities and properties, such as shape, warmth, or malleability that we perceive through our hands: When we touch, press, grasp, and squeeze a physical object we can feel how the object responds to our interaction through the perceived cutaneous and kinesthetic feedback. A rigid, flat surface, however, limits this perceived haptic richness to a minimum.

Force-based interaction could be interesting for future handheld devices that are made of flexible, shape-changing materials.

In the near future, however, handheld devices could be made from flexible, shape-changing materials that can then be both deformed by the user and by the device itself. Such flexible, actuated material could provide rich haptic feedback and open up a whole new category of force-based interaction techniques since the device physically responds to a user's touch, press, grasp, and squeeze. Think about a handheld device with a screen and a flexible body that the user can push into to move a 3D virtual object along the z-dimension. While the user feels how her finger is pushing into the soft body, the device could become rigid as soon as the virtual object collides with another object in the virtual scene. This way, the user actually feels the collision and cannot physically push any further.

Future handheld devices could sense force input on the entire device body.

Next, handheld devices could be enhanced to sense touch and force input around the whole device, i.e., at the front, back, sides, top, and bottom. This could make interaction with 3D virtual objects more natural, since the user can push the object along the x-axis by pushing at the sides, along the y-axis by pushing at the top or bottom, and move the object along the z-axis by pushing against the front or the BoD.

Wearable devices, which have a small input surface

Besides handheld devices, wearable devices could also benefit from force input. As we have seen in Chapter 7, our Force Picker enables value input at a very small gesture and display space.

Wearable devices, like smartwatches, only have a small surface for input and output. Hence, force-based interaction techniques would make ideal and efficient use of that small space.

could also benefit from force input.

8.3 Closing Remarks

The work presented in this thesis was inspired by the advent of Apple iPhone 6s and 6s Plus in September 2015—the first commercially available handheld devices with a force-sensitive capacitive touchscreen that continuously sense the intensity of every touch point at any location on the screen. In fact, all our interaction techniques presented in this thesis were tested on these force-sensitive touchscreens. Until 2019, Apple has released seven more iPhone models that feature force input, also called *3D Touch*.

Our work was inspired by Apple's announcement of their force-sensitive iPhones 6s and 6s Plus in September 2015.

Interestingly, Apple has omitted the force sensor in one of its latest models, iPhone Xr. Rumor has it that Apple is considering completely stepping away from *3D Touch*, and promote *Haptic Touch* instead [Rossignol, 2019]. Haptic Touch is a time-based touch gesture that is recognized when the user touches with her finger on the screen and keeps contact for about a second. Like for 3D Touch, feedback is given to the user through a short pulse signal from the vibration motor, called *Taptic Engine*, that is build into iPhone.

Apple tends to step back from 3D Touch, i.e., force input, on upcoming handheld devices.

One possible explanation for the potential shift from 3D Touch to Haptic Touch could be an undesired inconsistency of gesture use across handheld devices that support 3D Touch and those that cannot support it because the screen is not force-sensitive, like Apple's iPad tablet lineup. Although 3D Touch is way more expressive than Haptic Touch, Apple has always been artificially cutting it down: Currently, 3D Touch is only used to launch a preview mode or for opening context menus. Hence, the continuous range that force input provides is not fully exploited. While it is reasonable to keep the user experience as consistent as possible across force-sensitive and non-force-sensitive devices, Apple completely undermines the potential of force input for handheld interaction. Instead of assimilating 3D Touch to Haptic Touch,

For reasons of consistency to devices that do not have a force-sensitive touchscreen, such as iPad, Apple might favor *Haptic Touch*, i.e., time-based touch gestures, over force-based 3D Touch.

Discoverability of force input is key for a good user experience.

a maybe more desirable solution is providing a better user experience through discoverability of the two input techniques. In interfaces that respond to 3D Touch, it is often not evident for the user which UI elements react to 3D Touch and how these widgets respond to it. The same is true for Haptic Touch widgets. Understanding how to design for good discoverability is vital for a practical use of force input as interaction method. This, paired with a consistent, system-wide use of 3D Touch is essential for the user to understand when 3D Touch is available and when not.

At its current state, we consider force input an efficient extension to, but not a replacement for, existing touch input.

At its current state, we think that 3D Touch should be considered an addition to touch input that enables the user to interact more efficiently with iPhone. An apt example for this is scrolling through long lists, tables, or websites: On both, force-sensitive and non-sensitive devices, the user can scroll through these items with touch gestures, like swiping and flicking, at any time. However, on devices that provide force input, a more efficient scrolling based on rate-based force input, like ForceEdge [Antoine et al., 2017] or our Force Picker (cf. Chapter 7), could be provided. This way, force input brings additional benefits to users of force-sensitive iPhones, but maintains consistency with iPhones and iPads without a force-sensitive touchscreen: Scrolling via touch is still possible, yet slower.

We are looking forward to seeing new force-based interaction techniques in the near future that will further benefit interaction with handheld and wearable devices.

Bibliography

[1] Mandayam A. Srinivasan, Gerald L. Beauregard, David L. Brock. 1996. The Impact of Visual Information on Haptic Perception of Stiffness in Virtual Environment. *Proceedings of the ASME Dynamic Systems and Control Division*, 58, (January 1996).

[2] Motoyuki Akamatsu, I. Scott MacKenzie. 2000. Changes in Applied Force to a Touchpad During Pointing Tasks. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 44, 2, 359–359. DOI: 10.1177/154193120004400221. https://doi.org/10.1177/154193120004400221.

[3] Axel Antoine, Sylvain Malacria, Géry Casiez. 2017. ForceEdge: Controlling Autoscroll on Both Desktop and Mobile Computers Using the Force. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, Denver, Colorado, USA, 3281–3292. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025605. http://doi.acm.org/10.1145/3025453.3025605.

[4] Apple Inc. 2019. Apple Human Interface Guidelines. https://developer.apple.com/design/human-interface-guidelines/. Accessed May 17, 2019. (2019).

[5] Apple Inc. 2019. iOS 12 UIKit API Reference. https://developer.apple.com/reference/uikit/uitouch. Accessed May 17, 2019. (2019).

[6] Apple Inc. 2018. Taking Advantage of 3D Touch. https://developer.apple.com/ios/3d-touch/. Accessed May 17, 2019. (2018).

[7] Ahmed Sabbir Arif, Ali Mazalek, Wolfgang Stuerzlinger. 2014. The Use of Pseudo Pressure in Authenticating Smartphone Users. *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services* (MOBIQUITOUS '14). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), London, United Kingdom, 151–160. ISBN:

978-1-63190-039-6. DOI: 10.4108/icst.mobiquitous.2014.257919. http://dx.doi.org/10.4108/icst.mobiquitous.2014.257919.

[8] Ahmed Sabbir Arif, Wolfgang Stuerzlinger. 2013. Pseudo-Pressure Detection and Its Use in Predictive Text Entry on Touchscreens. *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration* (OzCHI '13). ACM, Adelaide, Australia, 383–392. ISBN: 978-1-4503-2525-7. DOI: 10.1145/2541016.2541024. http://doi.acm.org/10.1145/2541016.2541024.

[9] Mathias Baglioni, Sylvain Malacria, Eric Lecolinet, Yves Guiard. 2011. Flick-and-Brake: Finger Control over Inertial/Sustained Scroll Motion. *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '11). ACM, Vancouver, BC, Canada, 2281–2286. ISBN: 978-1-4503-0268-5. DOI: 10.1145/1979742.1979853. http://doi.acm.org/10.1145/1979742.1979853.

[10] Gary Barrett, Ryomei Omote. 2010. Projected-Capacitive Touch Technology. *Information Display*, 26, 3, 16–21.

[11] Patrick Baudisch, Gerry Chu. 2009. Back-of-Device Interaction Allows Creating Very Small Touch Devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, Boston, MA, USA, 1923–1932. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518995. http://doi.acm.org/10.1145/1518701.1518995.

[12] Hrvoje Benko, Andrew D. Wilson, Patrick Baudisch. 2006. Precise Selection Techniques for Multi-Touch Screens. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '06). ACM, Montreal, Quebec, Canada, 1263–1272. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124963. http://doi.acm.org/10.1145/1124772.1124963.

[13] Joanna Bergstrom-Lehtovirta, Antti Oulasvirta. 2014. Modeling the Functional Area of the Thumb on Mobile Touchscreen Surfaces. *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems* (CHI '14). ACM, Toronto, Ontario, Canada, 1991–2000. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557354. http://doi.acm.org/10.1145/2556288.2557354.

[14] Lonni Besançon, Mehdi Ammi, Tobias Isenberg. 2017. Pressure-Based Gain Factor Control for Mobile 3D Interaction Using Locally-Coupled Devices. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, Denver, Colorado, USA, 1831–1842. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025890. http://doi.acm.org/10.1145/3025453.3025890.

[15] Gábor Blaskó, Steven Feiner. 2004. Single-Handed Interaction Techniques for Multiple Pressure-Sensitive Strips. *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '04). ACM, Vienna, Austria, 1461–1464. ISBN: 1-58113-703-6. DOI: 10.1145/985921.986090. http://doi.acm.org/10.1145/985921.986090.

[16] Sebastian Boring, David Ledo, Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, Saul Greenberg. 2012. The Fat Thumb: Using the Thumb's Contact Size for Single-Handed Mobile Interaction. *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '12). ACM, San Francisco, California, USA, 39–48. ISBN: 978-1-4503-1105-2. DOI: 10.1145/2371574.2371582. http://doi.acm.org/10.1145/2371574.2371582.

[17] Stephen A. Brewster, Michael Hughes. 2009. Pressure-Based Text Entry for Mobile Devices. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '09) Article 9. ACM, Bonn, Germany, 9:1–9:4. ISBN: 978-1-60558-281-8. DOI: 10.1145/1613858.1613870. http://doi.acm.org/10.1145/1613858.1613870.

[18] Thorsten Büring, Jens Gerken, Harald Reiterer. 2008. Zoom Interaction Design for Pen-Operated Portable Devices. *International Journal of Human-Computer Studies*, 66, 8, (August 2008), 605–627. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2008.03.005. http://dx.doi.org/10.1016/j.ijhcs.2008.03.005.

[19] Daniel Buschek, Oliver Schoenleben, Antti Oulasvirta. 2014. Improving Accuracy in Back-of-Device Multitouch Typing: A Clustering-Based Approach to Keyboard Updating. *Proceedings of the 19th International Conference on Intelligent User Interfaces* (IUI '14). ACM, Haifa, Israel, 57–66. ISBN: 978-1-4503-2184-6. DOI: 10.1145/2557500.2557501. http://doi.acm.org/10.1145/2557500.2557501.

[20] William Buxton. 2016. Human Input to Computer Systems: Theories, Techniques and Technology; Chapter 2: An Illustrated Tour. https://www.billbuxton.com/input02.Devices.pdf. Accessed July 8, 2019]. (2016).

[21] William Buxton, Ralph Hill, Peter Rowley. 1985. Issues and Techniques in Touch-Sensitive Tablet Input. *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH '85). ACM, New York, NY, USA, 215–224. ISBN: 0-89791-166-0. DOI: 10.1145/325334.325239. http://doi.acm.org/10.1145/325334.325239.

[22] Stuart K. Card, William K. English, Betty J. Burr. 1978. Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a

CRT. *Ergonomics*, 21, 8, 601–613. DOI: 10.1080/00140137808931762. https://doi.org/10.1080/00140137808931762.

[23] Stuart K. Card, Jock D. Mackinlay, George G. Robertson. 1990. The Design Space of Input Devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '90). ACM, Seattle, Washington, USA, 117–124. ISBN: 0-201-50932-6. DOI: 10.1145/97243.97263. http://doi.acm.org/10.1145/97243.97263.

[24] Stuart Card, Thomas Moran, Allen Newell. 1986. The Model Human Processor: An Engineering Model of Human Performance. *Handbook of Perception and Human Performance*, 2, 45–1.

[25] Géry Casiez, Nicolas Roussel, Daniel Vogel. 2012. 1€ Filter: A Simple Speed-Based Low-Pass Filter for Noisy Input in Interactive Systems. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12). ACM, Austin, Texas, USA, 2527–2530. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208639. http://doi.acm.org/10.1145/2207676.2208639.

[26] Jared Cechanowicz, Pourang Irani, Sriram Subramanian. 2007. Augmenting the Mouse with Pressure Sensitive Input. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, San Jose, California, USA, 1385–1394. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240835. http://doi.acm.org/10.1145/1240624.1240835.

[27] Youli Chang, Sehi L'Yi, Kyle Koh, Jinwook Seo. 2015. Understanding Users' Touch Behavior on Large Mobile Touch-Screens and Assisted Targeting by Tilting Gesture. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, Seoul, Republic of Korea, 1499–1508. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702425. http://doi.acm.org/10.1145/2702123.2702425.

[28] Olivier Chapuis, Jean-Baptiste Labrune, Emmanuel Pietriga. 2009. DynaSpot: Speed-Dependent Area Cursor. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, Boston, MA, USA, 1391–1400. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518911. http://doi.acm.org/10.1145/1518701.1518911.

[29] Christian Corsten, Christian Cherek, Thorsten Karrer, Jan Borchers. 2015. HaptiCase: Back-of-Device Tactile Landmarks for Eyes-Free Absolute Indirect Touch. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, Seoul, Republic of Korea, 2171–2180. ISBN: 978-1-4503-

3145-6. DOI: 10.1145/2702123.2702277. http://doi.acm.org/10.1145/2702123.2702277.

[30] Christian Corsten, Bjoern Daehlmann, Simon Voelker, Jan Borchers. 2017. BackXPress: Using Back-of-Device Finger Pressure to Augment Touchscreen Input on Smartphones. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, Denver, Colorado, USA, 4654–4666. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025565. http://doi.acm.org/10.1145/3025453.3025565.

[31] Christian Corsten, Marcel Lahaye, Jan Borchers, Simon Voelker. 2019. ForceRay: Extending Thumb Reach via Force Input Stabilizes Device Grip for Mobile Touch Input. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI '19) Article 212. ACM, Glasgow, Scotland UK, 212:1–212:12. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300442. http://doi.acm.org/10.1145/3290605.3300442.

[32] Christian Corsten, Simon Voelker, Jan Borchers. 2017. Release, Don't Wait!: Reliable Force Input Confirmation with Quick Release. *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (ISS '17). ACM, Brighton, United Kingdom, 246–251. ISBN: 978-1-4503-4691-7. DOI: 10.1145/3132272.3134116. http://doi.acm.org/10.1145/3132272.3134116.

[33] Christian Corsten, Simon Voelker, Andreas Link, Jan Borchers. 2018. Use the Force Picker, Luke: Space-Efficient Value Input on Force-Sensitive Mobile Touchscreens. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18) Article 661. ACM, Montreal QC, Canada, 661:1–661:12. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174235. http://doi.acm.org/10.1145/3173574.3174235.

[34] Bjoern Daehlmann. 2016. *Pressure Input on the Back of Mobile Devices.* Master's Thesis. RWTH Aachen University, Aachen, Germany, (September 2016).

[35] Philip L. Davidson, Jefferson Y. Han. 2008. Extending 2D Object Arrangement with Pressure-Sensitive Layering Cues. *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (UIST '08). ACM, Monterey, CA, USA, 87–90. ISBN: 978-1-59593-975-3. DOI: 10.1145/1449715.1449730. http://doi.acm.org/10.1145/1449715.1449730.

[36] Jamie Davies. 2017. Infographic: What Do We Actually Use Our Smartphones for? http://telecoms.com/483334/infographic-what-do-we-actually-use-our-smartphones-for/. Accessed July 22, 2019. (2017).

[37] Alexander De Luca, Marian Harbach, Emanuel von Zezschwitz, Max-Emanuel Maurer, Bernhard Ewald Slawik, Heinrich Hussmann, Matthew Smith. 2014. Now You See Me, Now You Don't: Protecting Smartphone Authentication from Shoulder Surfers. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, Toronto, Ontario, Canada, 2937–2946. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557097. http://doi.acm.org/10.1145/2556288.2557097.

[38] N. I. Durlach, L. A. Delhorne, A. Wong, W. Y. Ko, W. M. Rabinowitz, J. Hollerbach. 1989. Manual Discrimination and Identification of Length by the Finger-Span Method. *Perception & Psychophysics*, 46, 1, 29–38. DOI: 10.3758/BF03208071. https://doi.org/10.3758/BF03208071.

[39] Rachel Eardley, Anne Roudaut, Steve Gill, Stephen J. Thompson. 2018. Designing for Multiple Hand Grips and Body Postures Within the UX of a Moving Smartphone. *Proceedings of the 2018 Designing Interactive Systems Conference* (DIS '18). ACM, Hong Kong, China, 611–621. ISBN: 978-1-4503-5198-0. DOI: 10.1145/3196709.3196711. http://doi.acm.org/10.1145/3196709.3196711.

[40] Rachel Eardley, Anne Roudaut, Steve Gill, Stephen J. Thompson. 2018. Investigating How Smartphone Movement Is Affected by Body Posture. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18) Article 202. ACM, Montreal QC, Canada, 202:1–202:8. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173776. http://doi.acm.org/10.1145/3173574.3173776.

[41] Rachel Eardley, Anne Roudaut, Steve Gill, Stephen J. Thompson. 2017. Understanding Grip Shifts: How Form Factors Impact Hand Movements on Mobile Phones. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, Denver, Colorado, USA, 4680–4691. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025835. http://doi.acm.org/10.1145/3025453.3025835.

[42] Kate Ehrlich. 1997. A Conversation with Ted Selker. *Interactions*, 4, 5, (September 1997), 34–47. ISSN: 1072-5520. DOI: 10.1145/264044.265483. http://doi.acm.org/10.1145/264044.265483.

[43] William English, Douglas Engelbart, Melvyn Berman. 1967. Display-Selection Techniques for Text Manipulation. *IEEE Transactions on Human Factors in Electronics*, 1, 21–31.

[44] Jon Fingas. 2012. Shocker: Smartphone Users Like Bigger Screebs, Market Share May Respond Accordingly. http://engadget.com/2012/09/03/shocker-smartphone-users-like-bigger-screens/. Accessed June 16, 2019. (2012).

[45] Paul M. Fitts. 1954. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology*, 47, 6, 381–391. DOI: http://dx.doi.org/10.1037/h0055392.

[46] Clifton Forlines, Chia Shen, William Buxton. 2005. Glimpse: A Novel Input Model for Multi-level Devices. *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '05). ACM, Portland, OR, USA, 1375–1378. ISBN: 1-59593-002-7. DOI: 10.1145/1056808.1056920. http://doi.acm.org/10.1145/1056808.1056920.

[47] Masitah Ghazali, Alan Dix. 2005. Knowledge of Today for the Design of Tomorrow. *Proceedings of the 2nd International Design and Engagibility Conference* (IDEC '05), 2–7. https://alandix.com/academic/papers/IDEC2005/design_principles-IDEC05.pdf.

[48] Mayank Goel, Jacob Wobbrock, Shwetak Patel. 2012. GripSense: Using Built-in Sensors to Detect Hand Posture and Pressure on Commodity Mobile Phones. *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (UIST '12). ACM, Cambridge, Massachusetts, USA, 545–554. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380184. http://doi.acm.org/10.1145/2380116.2380184.

[49] Alix Goguey, Sylvain Malacria, Carl Gutwin. 2018. Improving Discoverability and Expert Performance in Force-Sensitive Text Selection for Touch Devices with Mode Gauges. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18) Article 477. ACM, Montreal QC, Canada, 477:1–477:12. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174051. http://doi.acm.org/10.1145/3173574.3174051.

[50] Tovi Grossman, Ravin Balakrishnan. 2005. The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, Portland, Oregon, USA, 281–290. ISBN: 1-58113-998-5. DOI: 10.1145/1054972.1055012. http://doi.acm.org/10.1145/1054972.1055012.

[51] Hiroyuki Hakoda, Yoshitomo Fukatsu, Buntarou Shizuki, Jiro Tanaka. 2015. Back-of-Device Interaction Based on the Range of Motion of the Index Finger. *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction* (OzCHI '15). ACM, Parkville, VIC, Australia, 202–206. ISBN: 978-1-4503-3673-4. DOI: 10.1145/2838739.2838812. http://doi.acm.org/10.1145/2838739.2838812.

[52] Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, Roy Want. 1998. Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '98). ACM Press/Addison-Wesley Publishing Co., Los Angeles, California, USA, 17–24. ISBN: 0-201-30987-4. DOI: 10.1145/274644.274647. http://dx.doi.org/10.1145/274644.274647.

[53] Chris Harrison, Scott Hudson. 2012. Using Shear as a Supplemental Two-Dimensional Input Channel for Rich Touchscreen Interaction. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12). ACM, Austin, Texas, USA, 3149–3152. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208730. http://doi.acm.org/10.1145/2207676.2208730.

[54] Khalad Hasan, Junhyeok Kim, David Ahlström, Pourang Irani. 2016. Thumbs-Up: 3D Spatial Thumb-Reachable Space for One-Handed Thumb Interaction on Smartphones. *Proceedings of the 2016 Symposium on Spatial User Interaction* (SUI '16). ACM, Tokyo, Japan, 103–106. ISBN: 978-1-4503-4068-7. DOI: 10.1145/2983310.2985755. http://doi.acm.org/10.1145/2983310.2985755.

[55] H. Henningsen, S. Knecht, B. Ende-Henningsen. 1997. Influence of Afferent Feedback on Isometric Fine Force Resolution in Humans. *Experimental Brain Research*, 113, 2, (February 1997), 207–213. ISSN: 1432-1106. DOI: 10.1007/BF02450319. https://doi.org/10.1007/BF02450319.

[56] Henning Henningsen, Bettina Ende-Henningsen, Andrew M. Gordon. 1995. Contribution of Tactile Afferent Information to the Control of Isometric Finger Forces. *Experimental Brain Research*, 105, 2, 312–317. DOI: https://doi.org/10.1007/BF00240967.

[57] Seongkook Heo, Geehyuk Lee. 2011. Force Gestures: Augmenting Touch Screen Gestures with Normal and Tangential Forces. *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (UIST '11). ACM, Santa Barbara, California, USA, 621–626. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047278. http://doi.acm.org/10.1145/2047196.2047278.

[58] Seongkook Heo, Geehyuk Lee. 2012. ForceDrag: Using Pressure as a Touch Input Modifier. *Proceedings of the 24th Australian Computer-Human Interaction Conference* (OzCHI '12). ACM, Melbourne, Australia, 204–207. ISBN: 978-1-4503-1438-1. DOI: 10.1145/2414536.2414572. http://doi.acm.org/10.1145/2414536.2414572.

[59] Seongkook Heo, Geehyuk Lee. 2011. Forcetap: Extending the Input Vocabulary of Mobile Touch Screens by Adding Tap Gestures. *Proceedings of the 13th Interna-*

tional Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '11). ACM, Stockholm, Sweden, 113–122. ISBN: 978-1-4503-0541-9. DOI: 10.1145/2037373.2037393. http://doi.acm.org/10.1145/2037373.2037393.

[60] Christopher F. Herot, Guy Weinzapfel. 1978. One-Point Touch Input of Vector Information for Computer Displays. *SIGGRAPH Compututer Graphics*, 12, 3, (August 1978), 210–216. ISSN: 0097-8930. DOI: 10.1145/965139.807392. http://doi.acm.org/10.1145/965139.807392.

[61] Ken Hinckley, Mike Sinclair. 1999. Touch-Sensing Input Devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '99). ACM, Pittsburgh, Pennsylvania, USA, 223–230. ISBN: 0-201-48559-1. DOI: 10.1145/302979.303045. http://doi.acm.org/10.1145/302979.303045.

[62] Nambu Hirotaka. 2003. Reassessing Current Cell Phone Designs: Using Thumb Input Effectively. *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '03). ACM, Ft. Lauderdale, Florida, USA, 938–939. ISBN: 1-58113-637-4. DOI: 10.1145/765891.766081. http://doi.acm.org/10.1145/765891.766081.

[63] Bernhard Hirt, Harun Seyhan, Michael Wagner, Rainer Zumhasch. 2017. Hand and Wrist Anatomy and Biomechanics: A Comprehensive Guide. *European Journal of Orthopaedic Surgery & Traumatology*, 27, 7, (October 2017), 1029–1029. ISSN: 1432-1068. DOI: 10.1007/s00590-017-1991-z. https://doi.org/10.1007/s00590-017-1991-z.

[64] Eve Hoggan, Craig Stewart, Laura Haverinen, Giulio Jacucci, Vuokko Lantz. 2012. Pressages: Augmenting Phone Calls with Non-Verbal Messages. *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (UIST '12). ACM, Cambridge, Massachusetts, USA, 555–562. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380185. http://doi.acm.org/10.1145/2380116.2380185.

[65] David Holman, Andreas Hollatz, Amartya Banerjee, Roel Vertegaal. 2013. Unifone: Designing for Auxiliary Finger Input in One-Handed Mobile Interactions. *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction* (TEI '13). ACM, Barcelona, Spain, 177–184. ISBN: 978-1-4503-1898-3. DOI: 10.1145/2460625.2460653. http://doi.acm.org/10.1145/2460625.2460653.

[66] S. Lee Hong, Amber J. Brown, Karl M. Newell. 2008. Compensatory Properties of Visual Information in the Control of Isometric Force. *Perception & Psychophysics*,

70, 2, (February 2008), 306–313. ISSN: 1532-5962. DOI: 10.3758/PP.70.2.306. https://doi.org/10.3758/PP.70.2.306.

[67] Paul Horrell. 2014. Dash Forwards: The Evolution of Car Controls. https://360.here.com/2014/08/13/dash-forwards-evolution-car-controls/. Accessed July 23, 2019. (2014).

[68] Elizabeth Household, John McAuley, Thompson Charles, Timothy Lightfoot, Michael Swash. 2004. Analysis of Force Profile During a Maximum Voluntary Isometric Contraction Task. *Muscle & Nerve*, 29, 3, 401–408. DOI: 10.1002/mus.10564.

[69] Jochen Huber, Mohamed Sheik-Nainar, Nada Matic. 2017. Force-Enabled Touch Input on the Steering Wheel: An Elicitation Study. *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications Adjunct* (AutomotiveUI '17). ACM, Oldenburg, Germany, 168–172. ISBN: 978-1-4503-5151-5. DOI: 10.1145/3131726.3131740. http://doi.acm.org/10.1145/3131726.3131740.

[70] Sungjae Hwang, Andrea Bianchi, Kwang-Yun Wohn. 2013. VibPress: Estimating Pressure Input Using Vibration Absorption on Mobile Devices. *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '13). ACM, Munich, Germany, 31–34. ISBN: 978-1-4503-2273-7. DOI: 10.1145/2493190.2493193. http://doi.acm.org/10.1145/2493190.2493193.

[71] Sungjae Hwang, Kwang-yun Wohn. 2012. PseudoButton: Enabling Pressure-Sensitive Interaction by Repurposing Microphone on Mobile Device. *CHI '12 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '12). ACM, Austin, Texas, USA, 1565–1570. ISBN: 978-1-4503-1016-1. DOI: 10.1145/2212776.2223673. http://doi.acm.org/10.1145/2212776.2223673.

[72] Lynette A. Jones. 1989. Matching Forces: Constant Errors and Differential Thresholds. *Perception*, 18, 5, 681–687. DOI: 10.1068/p180681. https://doi.org/10.1068/p180681.

[73] Lynette A. Jones. 2000. Visual and Haptic Feedback in the Control of Force. *Experimental Brain Research*, 130, 2, (January 2000), 269–272. ISSN: 1432-1106. DOI: 10.1007/s002219900256. https://doi.org/10.1007/s002219900256.

[74] Lynette A. Jones, Ian W. Hunter. 1982. Force Sensation in Isometric Contractions: A Relative Force Effect. *Brain Research*, 244, 1, 186–189. ISSN: 0006-8993. DOI:

https://doi.org/10.1016/0006-8993(82)90919-2. http://www.sciencedirect.com/science/article/pii/0006899382909192.

[75] Lynette A. Jones, Ian W. Hunter. 1983. Perceived Force in Fatiguing Isometric Contractions. *Perception & Psychophysics*, 33, 4, 369–374. DOI: 10.3758/BF03205884. https://doi.org/10.3758/BF03205884.

[76] Lynette A. Jones, Erin Piateski. 2006. Contribution of Tactile Feedback from the Hand to the Perception of Force. *Experimental Brain Research*, 168, 1, (January 2006), 298–302. ISSN: 1432-1106. DOI: 10.1007/s00221-005-0259-8. https://doi.org/10.1007/s00221-005-0259-8.

[77] Amy K. Karlson, Benjamin B. Bederson. 2008. One-Handed Touchscreen Input for Legacy Applications. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, Florence, Italy, 1399–1408. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357274. http://doi.acm.org/10.1145/1357054.1357274.

[78] Amy K. Karlson, Benjamin B. Bederson. 2007. ThumbSpace: Generalized One-Handed Input for Touchscreen-Based Mobile Devices. *Proceedings of the 11th IFIP TC 13 International Conference on Human-Computer Interaction* (INTERACT '07). Springer-Verlag, Rio de Janeiro, Brazil, 324–338. ISBN: 3-540-74794-X, 978-3-540-74794-9. http://dl.acm.org/citation.cfm?id=1776994.1777034.

[79] Amy K. Karlson, Benjamin B. Bederson, Jose L. Contreras-Vidal. 2008. Understanding One-Handed Use of Mobile Devices. *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*. IGI Global, (January 2008), 86–101. DOI: 10.4018/978-1-59904-871-0.ch006.

[80] Sunjun Kim, Jihyun Yu, Geehyuk Lee. 2012. Interaction Techniques for Unreachable Objects on the Touchscreen. *Proceedings of the 24th Australian Computer-Human Interaction Conference* (OzCHI '12). ACM, Melbourne, Australia, 295–298. ISBN: 978-1-4503-1438-1. DOI: 10.1145/2414536.2414585. http://doi.acm.org/10.1145/2414536.2414585.

[81] Hiroshi Kinoshita, Peter R. Francis. 1996. A Comparison of Prehension Force Control in Young and Elderly Individuals. *European Journal of Applied Physiology and Occupational Physiology*, 74, 5, (November 1996), 450–460. ISSN: 1439-6327. DOI: 10.1007/BF02337726. https://doi.org/10.1007/BF02337726.

[82] Tracy L. Kivell. 2015. Evidence in Hand: Recent Discoveries and the Early Evolution of Human Manual Manipulation. *Philosophical Transactions of the Royal Society B:*

Biological Sciences, 370, 1682, 1–11. DOI: 10.1098/rstb.2015.0105. https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2015.0105.

[83] Akemi Kobayashi, Ryosuke Aoki, Norimichi Kitagawa, Toshitaka Kimura, Youichi Takashima, Tomohiro Yamada. 2016. Towards Enhancing Force-Input Interaction by Visual-Auditory Feedback ss an Introduction of First Use. *Proceedings, Part II, of the 18th International Conference on Human-Computer Interaction. Interaction Platforms and Techniques - Volume 9732*. Springer-Verlag, Berlin, Heidelberg, 180–191. ISBN: 978-3-319-39515-9. DOI: 10.1007/978-3-319-39516-6_17. https://doi.org/10.1007/978-3-319-39516-6_17.

[84] Takuro Kuribara, Buntarou Shizuki, Jiro Tanaka. 2015. Mouse Augmentation Using a Malleable Mouse Pad. *Human-Computer Interaction: Interaction Technologies*. Springer International Publishing, Cham, 217–226. ISBN: 978-3-319-20916-6. DOI: 10.1007/978-3-319-20916-6_21. https://doi.org/10.1007/978-3-319-20916-6_21.

[85] J. Lai, D. Zhang. 2015. ExtendedThumb: A Target Acquisition Approach for One-Handed Interaction with Touch-Screen Mobile Phones. *IEEE Transactions on Human-Machine Systems*, 45, 3, (June 2015), 362–370. ISSN: 2168-2291. DOI: 10.1109/THMS.2014.2377205.

[86] Huy Viet Le, Patrick Bader, Thomas Kosch, Niels Henze. 2016. Investigating Screen Shifting Techniques to Improve One-Handed Smartphone Usage. *Proceedings of the 9th Nordic Conference on Human-Computer Interaction* (NordiCHI '16) Article 27. ACM, Gothenburg, Sweden, 27:1–27:10. ISBN: 978-1-4503-4763-1. DOI: 10.1145/2971485.2971562. http://doi.acm.org/10.1145/2971485.2971562.

[87] S. K. Lee, William Buxton, Kenneth C. Smith. 1985. A Multi-Touch Three Dimensional Touch-Sensitive Tablet. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '85). ACM, San Francisco, California, USA, 21–25. ISBN: 0-89791-149-0. DOI: 10.1145/317456.317461. http://doi.acm.org/10.1145/317456.317461.

[88] Wing Ho Andy Li, Hongbo Fu. 2013. BezelCursor: Bezel-Initiated Cursor for One-Handed Target Acquisition on Mobile Touch Screens. *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications* (SA '13) Article 36. ACM, Hong Kong, Hong Kong, 36:1–36:1. ISBN: 978-1-4503-2633-9. DOI: 10.1145/2543651.2543680. http://doi.acm.org/10.1145/2543651.2543680.

[89] Yang Li, Ken Hinckley, Zhiwei Guan, James A. Landay. 2005. Experimental Analysis of Mode Switching Techniques in Pen-Based User Interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM,

Portland, Oregon, USA, 461–470. ISBN: 1-58113-998-5. DOI: 10.1145/1054972.1055036. http://doi.acm.org/10.1145/1054972.1055036.

[90] Chunyuan Liao, François Guimbretière, Corinna E. Loeckenhoff. 2006. Pen-Top Feedback for Paper-Based Interfaces. *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology* (UIST '06). ACM, Montreux, Switzerland, 201–210. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166285. http://doi.acm.org/10.1145/1166253.1166285.

[91] Andreas Link. 2017. *Bidirectional Force Input: Increasing and Decreasing Values on Mobile Devices with the Thumb*. Master's Thesis. RWTH Aachen University, Aachen, Germany, (September 2017).

[92] Markus Löchtefeld, Christoph Hirtz, Sven Gehring. 2013. Evaluation of Hybrid Front- and Back-of-Device Interaction on Mobile Devices. *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia* (MUM '13) Article 17. ACM, Lule, Sweden, 17:1–17:4. ISBN: 978-1-4503-2648-3. DOI: 10.1145/2541831.2541865. http://doi.acm.org/10.1145/2541831.2541865.

[93] Markus Löchtefeld, Phillip Schardt, Antonio Krüger, Sebastian Boring. 2015. Detecting Users' Handedness for Ergonomic Adaptation of Mobile User Interfaces. *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia* (MUM '15). ACM, Linz, Austria, 245–249. ISBN: 978-1-4503-3605-5. DOI: 10.1145/2836041.2836066. http://doi.acm.org/10.1145/2836041.2836066.

[94] I. Scott MacKenzie, Abigail Sellen, William A. S. Buxton. 1991. A Comparison of Input Devices in Element Pointing and Dragging Tasks. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '91). ACM, New Orleans, Louisiana, USA, 161–166. ISBN: 0-89791-383-3. DOI: 10.1145/108844.108868. http://doi.acm.org/10.1145/108844.108868.

[95] Jock Mackinlay, Stuart K. Card, George G. Robertson. 1990. A Semantic Analysis of the Design Space of Input Devices. *Human-Computer Interaction*, 5, 2, (June 1990), 145–190. ISSN: 0737-0024. DOI: 10.1207/s15327051hci0502\&3_2. https://doi.org/10.1207/s15327051hci0502%5C&3_2.

[96] Norbert Mai, Marc Avarello, Peter Bolsinger. 1985. Maintenance of Low Isometric Forces During Prehensile Grasping. *Neuropsychologia*, 23, 6, 805–812. ISSN: 0028-3932. DOI: https://doi.org/10.1016/0028-3932(85)90087-9. http://www.sciencedirect.com/science/article/pii/0028393285900879.

[97] David C. McCallum, Edward Mak, Pourang Irani, Sriram Subramanian. 2009. PressureText: Pressure Input for Mobile Phone Text Entry. *CHI '09 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '09). ACM, Boston, MA, USA, 4519–4524. ISBN: 978-1-60558-247-4. DOI: 10.1145/1520340.1520693. http://doi.acm.org/10.1145/1520340.1520693.

[98] Ross McLachlan, Daniel Boland, Stephen Brewster. 2014. Transient and Transitional States: Pressure as an Auxiliary Input Modality for Bimanual Interaction. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, Toronto, Ontario, Canada, 401–410. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557260. http://doi.acm.org/10.1145/2556288.2557260.

[99] Ross McLachlan, Stephen Brewster. 2015. Bimanual Input for Tablet Devices with Pressure and Multi-Touch Gestures. *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '15). ACM, Copenhagen, Denmark, 547–556. ISBN: 978-1-4503-3652-9. DOI: 10.1145/2785830.2785878. http://doi.acm.org/10.1145/2785830.2785878.

[100] Ross McLachlan, Stephen A. Brewster. 2013. Novel Modalities for Bimanual Scrolling on Tablet Devices. *Human-Computer Interaction – INTERACT 2013*. Springer Berlin Heidelberg, Berlin, Heidelberg, 229–246. ISBN: 978-3-642-40483-2. DOI: 10.1007/978-3-642-40483-2_16. https://doi.org/10.1007/978-3-642-40483-2_16.

[101] Margaret R. Minsky. 1984. Manipulating Simulated Objects with Real-World Gestures Using a Force and Position Sensitive Screen. *SIGGRAPH*, 18, 3, (January 1984), 195–203. ISSN: 0097-8930. DOI: 10.1145/964965.808598. http://doi.acm.org/10.1145/964965.808598.

[102] Anant Kartik Mithal, Sarah A. Douglas. 1996. Differences in Movement Microstructure of the Mouse and the Finger-Controlled Isometric Joystick. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '96). ACM, Vancouver, British Columbia, Canada, 300–307. ISBN: 0-89791-777-4. DOI: 10.1145/238386.238533. http://doi.acm.org/10.1145/238386.238533.

[103] Takashi Miyaki, Jun Rekimoto. 2009. GraspZoom: Zooming and Scrolling Control Model for Single-Handed Mobile Interaction. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '09) Article 11. ACM, Bonn, Germany, 11:1–11:4. ISBN: 978-1-60558-281-8. DOI: 10.1145/1613858.1613872. http://doi.acm.org/10.1145/1613858.1613872.

[104] Sachi Mizobuchi, Shinya Terasaki, Turo Keski-Jaskari, Jari Nousiainen, Matti Ryynanen, Miika Silfverberg. 2005. Making an Impression: Force-Controlled Pen Input for Handheld Devices. *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '05). ACM, Portland, OR, USA, 1661–1664. ISBN: 1-59593-002-7. DOI: 10.1145/1056808.1056991. http://doi.acm.org/10.1145/1056808.1056991.

[105] Mohammad Faizuddin Mohd Noor, Andrew Ramsay, Stephen Hughes, Simon Rogers, John Williamson, Roderick Murray-Smith. 2014. 28 Frames Later: Predicting Screen Touches from Back-of-Device Grip Changes. *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems* (CHI '14). ACM, Toronto, Ontario, Canada, 2005–2008. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557148. http://doi.acm.org/10.1145/2556288.2557148.

[106] Kevin R. Nelson. 2015. Exploring Apple's 3D Touch. https://tinyurl.com/y2uovc33. Accessed May 17, 2019. (2015).

[107] Karl M. Newell, P. Vernon McDonald. 1994. Information, Coordination Modes and Control in a Prehensile Force Task. *Human Movement Science*, 13, 3, 375–391. ISSN: 0167-9457. DOI: 10.1016/0167-9457(94)90046-9. http://www.sciencedirect.com/science/article/pii/0167945794900469.

[108] Alexander Ng, Stephen A. Brewster. 2016. Investigating Pressure Input and Haptic Feedback for In-Car Touchscreens and Touch Surfaces. *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (Automotive'UI 16). ACM, Ann Arbor, MI, USA, 121–128. ISBN: 978-1-4503-4533-0. DOI: 10.1145/3003715.3005420. http://doi.acm.org/10.1145/3003715.3005420.

[109] Alexander Ng, Stephen A. Brewster, Frank Beruscha, Wolfgang Krautter. 2017. An Evaluation of Input Controls for In-Car Interactions. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, Denver, Colorado, USA, 2845–2852. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025736. http://doi.acm.org/10.1145/3025453.3025736.

[110] Bernhard Niedermaier, Stephan Durach, Lutz Eckstein, Andreas Keinath. 2009. The New BMW iDrive – Applied Processes and Methods to Assure High Usability. *Digital Human Modeling*. Springer Berlin Heidelberg, 443–452. ISBN: 978-3-642-02809-0. DOI: https://doi.org/10.1007/978-3-642-02809-0_47.

[111] Nielsen Norman Group. 2015. Slider Design: Rules of Thumb. https://www.nngroup.com/articles/gui-slider-controls/. Accessed May 17, 2019. (2015).

[112] Donald A. Norman. 2002. *The Design of Everyday Things*. Basic Books, Inc., New York, NY, USA. ISBN: 9780465067107.

[113] Masaki Omata, Kenji Matsumura, Atsumi Imamiya. 2007. A Pressure-Sensing Mouse Button for Multilevel Click and Drag. *Human-Computer Interaction – INTERACT 2007*. Springer Berlin Heidelberg, Berlin, Heidelberg, 434–446. ISBN: 978-3-540-74796-3. DOI: 10.1007/978-3-540-74796-3_42. https://doi.org/10.1007/978-3-540-74796-3_42.

[114] OpenStax College. 2013. *Anatomy & Psychology*. OpenStax CNX. http://cnx.org/content/col11496/latest/.

[115] Xiao Dong Pang, Hong Z. Tan, Nathaniel I. Durlach. 1991. Manual Discrimination of Force Using Active Finger Motion. *Perception & Psychophysics*, 49, 6, (November 1991), 531–540. ISSN: 1532-5962. DOI: 10.3758/BF03212187. https://doi.org/10.3758/BF03212187.

[116] Dianne T. V. Pawluk, Robert D. Howe. 1999. Dynamic Contact of the Human Fingerpad Against a Flat Surface. *Journal of Biomechanical Engineering*, 121, 6, 605–611. DOI: 10.1115/1.2800860. http://dx.doi.org/10.1115/1.2800860.

[117] Sebastien Pelurson, Laurence Nigay. 2016. Bimanual Input for Multiscale Navigation with Pressure and Touch Gestures. *Proceedings of the 18th ACM International Conference on Multimodal Interaction* (ICMI '16). ACM, Tokyo, Japan, 145–152. ISBN: 978-1-4503-4556-9. DOI: 10.1145/2993148.2993152. http://doi.acm.org/10.1145/2993148.2993152.

[118] Siyuan Qiu, Lu Wang, Laikuan Wong. 2016. Pressure-Based Touch Positioning Techniques for 3D Objects. *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (I3D '16). ACM, Redmond, Washington, 199–200. ISBN: 978-1-4503-4043-4. DOI: 10.1145/2856400.2876010. http://doi.acm.org/10.1145/2856400.2876010.

[119] Philip Quinn. 2019. Estimating Touch Force with Barometric Pressure Sensors. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI '19) Article 689. ACM, Glasgow, Scotland UK, 689:1–689:7. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300919. http://doi.acm.org/10.1145/3290605.3300919.

[120] Philip Quinn, Andy Cockburn. 2009. Zoofing!: Faster List Selections with Pressure-Zoom-Flick-Scrolling. *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group* (OZCHI '09). ACM, Melbourne,

Australia, 185–192. ISBN: 978-1-60558-854-4. DOI: 10.1145/1738826.1738856. http://doi.acm.org/10.1145/1738826.1738856.

[121] Mahfuz Rahman, Sean Gustafson, Pourang Irani, Sriram Subramanian. 2009. Tilt Techniques: Investigating the Dexterity of Wrist-Based Input. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, Boston, MA, USA, 1943–1952. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518997. http://doi.acm.org/10.1145/1518701.1518997.

[122] Roope Raisamo. 1999. Evaluating Different Touch-Based Interaction Techniques in a Public Information Kiosk, (November 1999), 1–11.

[123] D. V. Raj, K. Ingty, M. S. Devanandan. 1985. Weight Appreciation in the Hand in Normal Subjects and in Patients with Leprous Neuropathy. *Brain*, 108, 1, (March 1985), 95–102. ISSN: 0006-8950. DOI: 10.1093/brain/108.1.95. https://doi.org/10.1093/brain/108.1.95.

[124] Gonzalo A. Ramos, Ravin Balakrishnan. 2007. Pressure Marks. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, San Jose, California, USA, 1375–1384. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240834. http://doi.acm.org/10.1145/1240624.1240834.

[125] Gonzalo Ramos, Ravin Balakrishnan. 2005. Zliding: Fluid Zooming and Sliding for High Precision Parameter Manipulation. *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (UIST '05). ACM, Seattle, WA, USA, 143–152. ISBN: 1-59593-271-2. DOI: 10.1145/1095034.1095059. http://doi.acm.org/10.1145/1095034.1095059.

[126] Gonzalo Ramos, Matthew Boulos, Ravin Balakrishnan. 2004. Pressure Widgets. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, Vienna, Austria, 487–494. ISBN: 1-58113-702-8. DOI: 10.1145/985692.985754. http://doi.acm.org/10.1145/985692.985754.

[127] Jef Raskin. 2000. *The Humane Interface: New Directions for Designing Interactive Systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. ISBN: 0-201-37937-6.

[128] Jun Rekimoto, Carsten Schwesig. 2006. PreSenseII: Bi-Directional Touch and Pressure Sensing Interactions with Tactile Feedback. *CHI '06 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '06). ACM, Montreal, Quebec, Canada, 1253–1258. ISBN: 1-59593-298-4. DOI: 10.1145/1125451.1125685. http://doi.acm.org/10.1145/1125451.1125685.

[129] Xiangshi Ren, Jibin Yin, Shengdong Zhao, Yang Li. 2007. The Adaptive Hybrid Cursor: A Pressure-Based Target Selection Technique for Pen-Based User Interfaces. *Proceedings of the 11th IFIP TC 13 International Conference on Human-Computer Interaction* (INTERACT' 07). Springer-Verlag, Rio de Janeiro, Brazil, 310–323. ISBN: 3-540-74794-X, 978-3-540-74794-9. http://dl.acm.org/citation.cfm?id=1776994.1777033.

[130] Christian Rendl, Patrick Greindl, Michael Haller, Martin Zirkl, Barbara Stadlober, Paul Hartmann. 2012. PyzoFlex: Printed Piezoelectric Pressure Sensing Foil. *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (UIST '12). ACM, Cambridge, Massachusetts, USA, 509–518. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380180. http://doi.acm.org/10.1145/2380116.2380180.

[131] Christian Rendl, David Kim, Patrick Parzer, Sean Fanello, Martin Zirkl, Gregor Scheipl, Michael Haller, Shahram Izadi. 2016. FlexCase: Enhancing Mobile Interaction with a Flexible Sensing and Display Cover. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, San Jose, California, USA, 5138–5150. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858314. http://doi.acm.org/10.1145/2858036.2858314.

[132] Simon Rogers, John Williamson, Craig Stewart, Roderick Murray-Smith. 2011. AnglePose: Robust, Precise Capacitive Touch Tracking via 3D Orientation Estimation. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, Vancouver, BC, Canada, 2575–2584. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979318. http://doi.acm.org/10.1145/1978942.1979318.

[133] Ilya Rosenberg, Ken Perlin. 2009. The UnMousePad: An Interpolating Multi-Touch Force-Sensing Input Pad. *ACM Transations on Graphics*, 28, 3, Article 65, (July 2009), 65:1–65:9. ISSN: 0730-0301. DOI: 10.1145/1531326.1531371. http://doi.acm.org/10.1145/1531326.1531371.

[134] Joe Rossignol. 2019. Apple Expected to Remove 3D Touch from All 2019 iPhones in Favor of Haptic Touch. https://www.macrumors.com/2019/05/27/no-3d-touch-2019-iphones-removed-rumor/. Accessed May 27, 2019. (2019).

[135] Anne Roudaut, Stéphane Huot, Eric Lecolinet. 2008. TapTap and MagStick: Improving One-Handed Target Acquisition on Small Touch-Screens. *Proceedings of the Working Conference on Advanced Visual Interfaces* (AVI '08). ACM, Napoli, Italy, 146–153. ISBN: 978-1-60558-141-5. DOI: 10.1145/1385569.1385594. http://doi.acm.org/10.1145/1385569.1385594.

[136] Anne Roudaut, Eric Lecolinet, Yves Guiard. 2009. MicroRolls: Expanding Touch-Screen Input Vocabulary by Distinguishing Rolls vs. Slides of the Thumb. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, Boston, MA, USA, 927–936. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518843. http://doi.acm.org/10.1145/1518701.1518843.

[137] Joseph D. Rutledge, Ted Selker. 1990. Force-to-Motion Functions for Pointing. *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction* (INTERACT '90). North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 701–706. ISBN: 0-444-88817-9. http://dl.acm.org/citation.cfm?id=647402.725310.

[138] Lawrence Sambrooks, Brett Wilkinson. 2013. Comparison of Gestural, Touch, and Mouse Interaction with Fitts' Law. *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration* (OzCHI '13). ACM, Adelaide, Australia, 119–122. ISBN: 978-1-4503-2525-7. DOI: 10.1145/2541016.2541066. http://doi.acm.org/10.1145/2541016.2541066.

[139] L. Sasaki, G. Fedorkow, William Buxton, Chris Retterath, K. C. Smith. 1981. A Touch-Sensitive Input Device, (January 1981), 293–279. http://hdl.handle.net/2027/spo.bbp2372.1981.037.

[140] Oliver Schoenleben, Antti Oulasvirta. 2013. Sandwich Keyboard: Fast Ten-Finger Typing on a Mobile Device with Adaptive Touch Sensing on the Back Side. *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '13). ACM, Munich, Germany, 175–178. ISBN: 978-1-4503-2273-7. DOI: 10.1145/2493190.2493233. http://doi.acm.org/10.1145/2493190.2493233.

[141] Carsten Schwesig, Ivan Poupyrev, Eijiro Mori. 2004. Gummi: A Bendable Computer. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, Vienna, Austria, 263–270. ISBN: 1-58113-702-8. DOI: 10.1145/985692.985726. http://doi.acm.org/10.1145/985692.985726.

[142] Ted Selker, J. Rutledge. 1991. Finger Force Precision for Computer Pointing. *IBM Research*, 17342, 2–7.

[143] Mohamed Sheik-Nainar, Jochen Huber, Raja Bose, Nada Matic. 2016. Force-Enabled TouchPad in Cars: Improving Target Selection Using Absolute Input. *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (CHI EA '16). ACM, San Jose, California, USA, 2697–2704. ISBN: 978-1-4503-

4082-3. DOI: 10.1145/2851581.2892390. http://doi.acm.org/10.1145/2851581.2892390.

[144] Erh-li Early Shen, Sung-sheng Daniel Tsai, Hao-hua Chu, Yung-jen Jane Hsu, Chi-wen Euro Chen. 2009. Double-Side Multi-Touch Input for Mobile Devices. *CHI '09 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '09). ACM, Boston, MA, USA, 4339–4344. ISBN: 978-1-60558-247-4. DOI: 10.1145/1520340.1520663. http://doi.acm.org/10.1145/1520340.1520663.

[145] Kang Shi, Pourang Irani, Sean Gustafson, Sriram Subramanian. 2008. PressureFish: A Method to Improve Control of Discrete Pressure-Based Input. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, Florence, Italy, 1295–1298. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357256. http://doi.acm.org/10.1145/1357054.1357256.

[146] Kang Shi, Sriram Subramanian, Pourang Irani. 2009. PressureMove: Pressure Input with Mouse Movement. *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II* (INTERACT '09). Springer-Verlag, Uppsala, Sweden, 25–39. ISBN: 978-3-642-03657-6. DOI: 10.1007/978-3-642-03658-3_7. https://doi.org/10.1007/978-3-642-03658-3_7.

[147] Ben Shneiderman. 1997. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* (3rd edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN: 0201694972.

[148] Katie A. Siek, Yvonne Rogers, Kay H. Connelly. 2005. Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs. *Proceedings of the 2005 IFIP TC13 International Conference on Human-Computer Interaction* (INTERACT '05). Springer-Verlag, Rome, Italy, 267–280. ISBN: 3-540-28943-7, 978-3-540-28943-2. DOI: 10.1007/11555261_24. http://dx.doi.org/10.1007/11555261_24.

[149] Andrew B. Slifkin, Karl M. Newell. 1999. Noise, Information Transmission, and Force Variability. *Journal of Experimental Psychology: Human Perception and Performance*, 25, 3, 837. DOI: 10.1037/0096-1523.25.3.837.

[150] Jacob J. Sosnoff, Karl M. Newell. 2005. Intermittent Visual Information and the Multiple Time Scales of Visual Motor Control of Continuous Isometric Force Production. *Perception & Psychophysics*, 67, 2, (February 2005), 335–344. ISSN: 1532-5962. DOI: 10.3758/BF03206496. https://doi.org/10.3758/BF03206496.

[151] R. William Soukoreff, I. Scott MacKenzie. 2004. Towards a Standard for Pointing Device Evaluation, Perspectives on 27 Years of Fitts' Law Research in HCI. *International Journal of Human-Compututer Studies*, 61, 6, (December 2004), 751–789.

ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2004.09.001. http://dx.doi.org/10.1016/j.ijhcs.2004.09.001.

[152] Caleb Southern, James Clawson, Brian Frey, Gregory Abowd, Mario Romero. 2012. Braille Touch: Mobile Touchscreen Text Entry for the Visually Impaired. *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services Companion* (MobileHCI '12). ACM, San Francisco, California, USA, 155–156. ISBN: 978-1-4503-1443-5. DOI: 10.1145/2371664.2371696. http://doi.acm.org/10.1145/2371664.2371696.

[153] Daniel Spelmezan, Caroline Appert, Olivier Chapuis, Emmanuel Pietriga. 2013. Controlling Widgets with One Power-Up Button. *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (UIST '13). ACM, St. Andrews, Scotland, United Kingdom, 71–74. ISBN: 978-1-4503-2268-3. DOI: 10.1145/2501988.2502025. http://doi.acm.org/10.1145/2501988.2502025.

[154] Daniel Spelmezan, Caroline Appert, Olivier Chapuis, Emmanuel Pietriga. 2013. Side Pressure for Bidirectional Navigation on Small Devices. *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '13). ACM, Munich, Germany, 11–20. ISBN: 978-1-4503-2273-7. DOI: 10.1145/2493190.2493199. http://doi.acm.org/10.1145/2493190.2493199.

[155] Allison Stadd. 2013. 79% Of People 18-44 Have Their Smartphones with Them 22 Hours a Day – A Study. https://www.adweek.com/digital/smartphones/. Accessed July 22, 2019. (2013).

[156] 2011. *Tremor and Clonus. Comprehensive Physiology*. American Cancer Society, 325–343. ISBN: 9780470650714. DOI: 10.1002/cphy.cp010209. https://onlinelibrary.wiley.com/doi/abs/10.1002/cphy.cp010209.

[157] Craig Stewart, Eve Hoggan, Laura Haverinen, Hugues Salamin, Giulio Jacucci. 2012. An Exploration of Inadvertent Variations in Mobile Pressure Input. *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '12). ACM, San Francisco, California, USA, 35–38. ISBN: 978-1-4503-1105-2. DOI: 10.1145/2371574.2371581. http://doi.acm.org/10.1145/2371574.2371581.

[158] Craig Stewart, Michael Rohs, Sven Kratz, Georg Essl. 2010. Characteristics of Pressure-Based Input for Mobile Devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, Atlanta, Georgia, USA,

801–810. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753444. http://doi.acm.org/10.1145/1753326.1753444.

[159] Qingkun Su, Oscar Kin-Chung Au, Pengfei Xu, Hongbo Fu, Chiew-Lan Tai. 2016. 2D-Dragger: Unified Touch-Based Target Acquisition with Constant Effective Width. *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '16). ACM, Florence, Italy, 170–179. ISBN: 978-1-4503-4408-1. DOI: 10.1145/2935334.2935339. http://doi.acm.org/10.1145/2935334.2935339.

[160] Faisal Taher, Jason Alexander, John Hardy, Eduardo Velloso. 2014. An Empirical Characterization of Touch-Gesture Input-Force on Mobile Devices. *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (ITS '14). ACM, Dresden, Germany, 195–204. ISBN: 978-1-4503-2587-5. DOI: 10.1145/2669485.2669515. http://doi.acm.org/10.1145/2669485.2669515.

[161] Ryosuke Takada, Toshiyuki Ando, Buntarou Shizuki, Shin Takahashi. 2019. Baro-Touch: A Technique for Touch Force Sensing Using a Waterproof Device's Built-in Barometer. *Journal of Information Processing*, 27, 106–115. DOI: 10.2197/ipsjjip.27.106.

[162] Hong Z. Tan, Xiao Dong Pang, Nathaniel I. Durlach. 1992. Manual Resolution of Length, Force, and Compliance. *Advances in Robotics*, 42, 13–18.

[163] Hsin-Ruey Tsai, Da-Yuan Huang, Chen-Hsin Hsieh, Lee-Ting Huang, Yi-Ping Hung. 2016. MovingScreen: Selecting Hard-to-Reach Targets with Automatic Comfort Zone Calibration on Mobile Devices. *Adjunct Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '16). ACM, Florence, Italy, 651–658. ISBN: 978-1-4503-4413-5. DOI: 10.1145/2957265.2961835. http://doi.acm.org/10.1145/2957265.2961835.

[164] Bret Victor. 2011. A Brief Rant on the Future of Interaction Design. http://worrydream.com/ABriefRantOnTheFutureOfInteractionDesign/. Accessed May 17, 2019. (2011).

[165] Feng Wang, Xiang Cao, Xiangshi Ren, Pourang Irani. 2009. Detecting and Leveraging Finger Orientation for Interaction with Direct-Touch Surfaces. *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (UIST '09). ACM, Victoria, BC, Canada, 23–32. ISBN: 978-1-60558-745-5. DOI: 10.1145/1622176.1622182. http://doi.acm.org/10.1145/1622176.1622182.

[166] Feng Wang, Xiangshi Ren. 2009. Empirical Evaluation for Finger Input Properties in Multi-Touch Interaction. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, Boston, MA, USA, 1063–1072. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518864. http://doi.acm.org/10.1145/1518701.1518864.

[167] Yoichi Watanabe, Yasutoshi Makino, Katsunari Sato, Takashi Maeno. 2012. Contact Force and Finger Angles Estimation for Touch Panel by Detecting Transmitted Light on Fingernail. *Haptics: Perception, Devices, Mobility, and Communication.* Springer Berlin Heidelberg, Berlin, Heidelberg, 601–612. ISBN: 978-3-642-31401-8. DOI: 10.1007/978-3-642-31401-8_53. https://doi.org/10.1007/978-3-642-31401-8_53.

[168] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, Chia Shen. 2007. Lucid Touch: A See-Through Mobile Device. *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology* (UIST '07). ACM, Newport, Rhode Island, USA, 269–278. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294259. http://doi.acm.org/10.1145/1294211.1294259.

[169] Daniel Wigdor, Dennis Wixon. 2011. *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture.* Elsevier. ISBN: 978-0-12-382231-4. DOI: 10.1016/C2009-0-64091-5. https://doi.org/10.1016/C2009-0-64091-5.

[170] Graham Wilson, Stephen A. Brewster, Martin Halvey. 2011. The Effects of Walking and Control Method on Pressure-Based Interaction. *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '11). ACM, Vancouver, BC, Canada, 2275–2280. ISBN: 978-1-4503-0268-5. DOI: 10.1145/1979742.1979943. http://doi.acm.org/10.1145/1979742.1979943.

[171] Graham Wilson, Stephen Brewster, Martin Halvey. 2013. Towards Utilising One-Handed Multi-Digit Pressure Input. *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '13). ACM, Paris, France, 1317–1322. ISBN: 978-1-4503-1952-2. DOI: 10.1145/2468356.2468591. http://doi.acm.org/10.1145/2468356.2468591.

[172] Graham Wilson, David Hannah, Stephen Brewster, Martin Halvey. 2012. Investigating One-Handed Multi-Digit Pressure Input for Mobile Devices. *CHI '12 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '12). ACM, Austin, Texas, USA, 1727–1732. ISBN: 978-1-4503-1016-1. DOI: 10.1145/2212776.2223700. http://doi.acm.org/10.1145/2212776.2223700.

[173] Graham Wilson, Craig Stewart, Stephen A. Brewster. 2010. Pressure-Based Menu Selection for Mobile Devices. *Proceedings of the 12th International Conference*

on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '10). ACM, Lisbon, Portugal, 181–190. ISBN: 978-1-60558-835-3. DOI: 10.1145/1851600.1851631. http://doi.acm.org/10.1145/1851600.1851631.

[174] Katrin Wolf, Christian Müller-Tomfelde, Kelvin Cheng, Ina Wechsung. 2012. PinchPad: Performance of Touch-Based Gestures While Grasping Devices. *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction* (TEI '12). ACM, Kingston, Ontario, Canada, 103–110. ISBN: 978-1-4503-1174-8. DOI: 10.1145/2148131.2148155. http://doi.acm.org/10.1145/2148131.2148155.

[175] Xing-Dong Yang, Edward Mak, Pourang Irani, Walter F. Bischof. 2009. Dual-Surface Input: Augmenting One-Handed Interaction with Coordinated Front and Behind-the-Screen Input. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '09) Article 5. ACM, Bonn, Germany, 5:1–5:10. ISBN: 978-1-60558-281-8. DOI: 10.1145/1613858.1613865. http://doi.acm.org/10.1145/1613858.1613865.

[176] Hyunjin Yoo, Jungwon Yoon, Hyunsoo Ji. 2015. Index Finger Zone: Study on Touchable Area Expandability Using Thumb and Index Finger. *Adjunct Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '15). ACM, Copenhagen, Denmark, 803–810. ISBN: 978-1-4503-3653-6. DOI: 10.1145/2786567.2793704. http://doi.acm.org/10.1145/2786567.2793704.

[177] Neng-Hao Yu, Da-Yuan Huang, Jia-Jyun Hsu, Yi-Ping Hung. 2013. Rapid Selection of Hard-to-Access Targets by Thumb on Mobile Touch-Screens. *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '13). ACM, Munich, Germany, 400–403. ISBN: 978-1-4503-2273-7. DOI: 10.1145/2493190.2493202. http://doi.acm.org/10.1145/2493190.2493202.

[178] Shumin Zhai, Barton A. Smith, Ted Selker. 1997. Dual Stream Input for Pointing and Scrolling. *CHI '97 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '97). ACM, Atlanta, Georgia, 305–306. ISBN: 0-89791-926-2. DOI: 10.1145/1120212.1120406. http://doi.acm.org/10.1145/1120212.1120406.

[179] Cheng Zhang, Anhong Guo, Dingtian Zhang, Caleb Southern, Rosa Arriaga, Gregory Abowd. 2015. BeyondTouch: Extending the Input Language with Built-in Sensors on Commodity Smartphones. *Proceedings of the 20th International Conference on Intelligent User Interfaces* (IUI '15). ACM, Atlanta, Georgia, USA, 67–77. ISBN: 978-1-4503-3306-1. DOI: 10.1145/2678025.2701374. http://doi.acm.org/10.1145/2678025.2701374.

[180] Mingyuan Zhong, Chun Yu, Qian Wang, Xuhai Xu, Yuanchun Shi. 2018. ForceBoard: Subtle Text Entry Leveraging Pressure. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18) Article 528. ACM, Montreal QC, Canada, 528:1–528:10. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174102. http://doi.acm.org/10.1145/3173574.3174102.

Index

1-Tap, 92, 98–100, 102, 105
1€ Filter, 153
2D-Dragger, 140
3D Touch, 5, 140, 201, 202

absolute mapping, 25
acceleration, 12
accelerometer, 37
acoustic wave sensing, 52
Analog-to-Digital Converter (ADC), 28
Android, 164, 172–174
AnglePose, 152
applied force, 12
Arduino, 28
area (A), 12

back-of-device (BoD), 85–117, 139
back-of-device interaction (BoDI), 86
BackXPress, 9, 85–117, 196, 199
barometer, 38
BeyondTouch, 89
BezelCursor, 135, 138–140, 144, 161
BezelSpace, 140, 156
Bluetooth, 115, 116
bounce back, 91
BrailleTouch, 89
Bubble Cursor, 142

capacitance, 31, 55
capacitor, 31
Carpal Bones, 15
Carpal Tunnel, 15
Carpometacarpal Joint (CMC), 141, 142, 145, 151, 159
Cathode Ray Tube (CRT), 52
CMN model, 123, 127, 132, 133
confirmation technique, 44, 47
Contact Event, 112
contact force, 12

contactless force, 12
control mechanism, 47
control-gain ratio, 72
CornerSpace, 140, 156
Cursor Technique, 139
cutaneous feedback, 20–22

decision tree, 37, 159, 160
design space, 53, 167
digitizer, 30, 31, 52
direct manipulation, 1
display footprint, 164
Distal Interphalangeal Joint (DIP), 15
Distal Phalanx, 15, 16
dominant hand (DH), 64, 69, 72
Dorsal Interossei Muscles, 15, 17
Dual-Surface Input, 140
Dwell Time, 44, 45, 47, 56, 91, 119–134
dynamometer, 13, 20
DynaSpot, 142, 144, 146

ExtendedThumb, 140
Extendible Cursor, 140
external feedback, 20, 21, 23
extrinsic muscle, 14, 15

Fast Fourier Transformation, 37
fingernail sensor, 38
fisheye function, 46, 51
Fitts' Law, 35, 36, 60
FlexCase, 65, 68
Flexor Digitorium Profundus, 16, 17
Flexor Digitorum Superficialis, 16, 17
Flexor Pollicis Longus, 16, 17
force (F), 11, 12
force history, 115
force lock, 100, 114
force meter, 13
Force Picker, 9, 198, 199
Force Pulse, 160, 170
Force Sensing Resistor (FSR), 27, 28, 69, 71, 115, 116
force space, 42
force touch, 141
ForceBoard, 140
ForceDrag, 89
ForceEdge, 140, 202
ForcePicker, 163–193
ForceRay, 9, 135–162, 197, 199
Forcetap, 89

gain factor, 73
gesture footprint, 147, 151, 152, 155, 164, 179, 180, 183, 185, 186, 188
graphic tablet, 56
GraspZoom, 168, 169
grip change, 147, 151, 155
gyroscope, 36, 140

Haptic Touch, 201
HaptiCase, 89
harpsichord, 2
high pass filter, 37
Hooke's Law, 13
HTC U11, 165
Huawei Mate S, 165
Human-Computer Interaction (HCI), 6, 49
Hypothenar Muscles, 15, 17
hysteresis, 42

inadvertent force, 25
indium thin oxide (ITO), 31
internal feedback, 20–22
Interpolating Force Sensing Resistance (IFSR), 27
intrinsic muscle, 14, 15
iOS, 125, 138, 143, 144, 158–160, 172–174, 182, 191, 193
iOS Control Center, 189
iOS Home Screen, 147
iOS SDK, 35, 96, 144
iPad, 52, 201
iPhone 6, 95
iPhone 6 Plus, 172
iPhone 6s, 31, 33, 95, 123, 132, 201
iPhone 6s Plus, 143, 172, 201
iPhone 7 Plus, 172
iPhone 8, 164
iPhone 8 Plus, 172
iPhone X, 158
iPhone XR, 201
iPhone Xs, 86, 120, 122, 136, 158
isometric, 16
isometric joystick, 3, 60, 62
isotonic, 16

Just Noticeable Difference (JND), 19

kinesthetic feedback, 20–22

liquid crystal display (LCD), 31
low pass filter, 37
LucidTouch, 88
Lumbricalis Muscles, 16, 17

macOS Dock, 64, 65
MagStick, 64, 138, 140, 144
mass (m), 12
Maximum Voluntary Contraction (MVC), 18
Metacarpal Bones, 15
Metacarpophalangeal Joint (MCP), 15
microcontroller, 28, 115
MicroRolls, 171
Middle Phalanx, 15, 16
minimized Thumb-Roll Picker, 182–188, 192, 199
MovingScreen, 139

N-Tap, 92, 98, 100, 106, 107, 110
NanoTouch, 88
natural inverse, 44, 91
natural mapping, 42, 173
Newton (N), 12
Newton's Three Laws of Motion, 11
non-dominant hand (NDH), 64, 70, 72

Ohm's Law, 28
One-Handed Mode, 138, 139, 144
organic light-emitting diode (OLED), 28, 30

Palmar Interossei Muscles, 15, 17
Pascal (pa), 12
Personal Digital Assistant (PDA), 20, 64, 69, 171
phalanx, 15, 16
piano, 2, 4, 5
piezoelectric effect, 32
Pointing Stick, 60
polar coordinate, 142
Position-Based Control, 47, 48, 63, 142, 165, 167
Positional Control, *see* Position-Based Control
potentiometer, 29
PreSense II, 168, 169
Press-Through, 170
Pressages, 69
pressure (P), 12, 38
Pressure Mark, 58, 59
pressure space, 42
Pressure Widgets, 57
Pressure-Based Linear Targeting (PBLT), 41, 43, 56, 69
Pressure.js, 36
Proximal Interphalangeal Joint (PIP), 15
Proximal Phalanx, 15, 16
Proxy Region Technique, 139
pseudo force, 26
pulling force, 13
PyzoFlex, 32

quasi-mode, 86, 92, 137, 171
Quick Release, 44, 45, 47, 56, 119–134, 142, 145, 197, 199

Rate-Based Control, 47, 48, 63, 165, 167–169
relative mapping, 25
resting threshold, 141, 169
restoring force, 13

Samsung Galaxy S5, 164
Screen Transform Technique, 138–140
shear force, 66, 71, 121
sigmoid function, 46, 90
Sliding Screen, 139
Spaceball, 5, 7
strain gauge, 4, 20, 52, 53, 60
System Picker, 173, 198

tangential force, *see* shear force
Tap-and-Refine, 51
TapTap, 139
Taptic Engine, 33, 201
Thenar Muscles, 16, 17
Thresholding, 44
Thumb-Roll, 170, 171, 192, 198
ThumbSpace, 139
TiltCursor, 139
TiltReduction, 139
TiltSlide, 139
touch ellipsis, 33
TrackPoint, 6, 18, 61
transfer function, 46, 90
transient, 86
tremor, 61

UnMousePad, 28

Velocity-Based Control, *see* Rate-Based Control
vibration, 37
vibrotactile feedback, 24
voltage divider, 28

Windows, Icons, Menus, Pointers (WIMP), 50

Own Publications

Papers (Peer-reviewed, archival)

Simon Voelker, Sebastian Hueber, **Christian Corsten**, Christian Remy. 2020. HeadReach: Using Head Tracking to Increase Reachability on Mobile Touch Devices. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (CHI '20) Article 739. ACM, Honolulu Hawai'i, USA, 739:1–739:12. ISBN: 978-1-4503-6708-0/20/04. DOI: 10.1145/3313831.3376868. https://doi.org/10.1145/3313831.3376868.

Contribution and Benefits: Compares head tracking-based techniques to address reachability issues for one-handed smartphone use. Two of the techniques yield higher success rates for selecting targets compared to direct touch input with the thumb.

Acceptance Rate: 24%.

Papers (Peer-reviewed, archival)

Christian Corsten, Marcel Lahaye, Jan Borchers, Simon Voelker. 2019. ForceRay: Extending Thumb Reach via Force Input Stabilizes Device Grip for Mobile Touch Input. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI '19) Article 212. ACM, Glasgow, Scotland, UK, 212:1–212:12. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300442. http://doi.acm.org/10.1145/3290605.3300442.

Contribution and Benefits: Presents ForceRay, a solution to reachability issues for one-handed smartphone interaction using force input. Enables to select targets at edges and corners quickly while maintaining a stable device grip.

Acceptance Rate: 24%.

Christian Corsten, Simon Voelker, Andreas Link, Jan Borchers. 2018. Use the Force Picker, Luke: Space-Efficient Value Input on Force-Sensitive Mobile Touchscreens. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18) Article 661. ACM, Montreal QC, Canada, 661:1–661:12. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174235. http://doi.acm.org/10.1145/3173574.3174235.

Contribution and Benefits: Value pickers on smartphones occupy significant gesture and display space. Our Force Picker, instead, allows for bidirectional force control to select values. Requires no more space than that of the initial touch point.

★ *Honorable Mention Award*

Acceptance Rate: 26%.

Christian Corsten, Simon Voelker, Jan Borchers. 2017. Release, Don't Wait!: Reliable Force Input Confirmation with Quick Release. *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (ISS '17). ACM, Brighton, United Kingdom, 246–251. ISBN: 978-1-4503-4691-7. DOI: 10.1145/3132272.3134116. http://doi.acm.org/10.1145/3132272.3134116.

Contribution and Benefits: Presents an algorithm to detect Quick Release, a technique to confirm force input. Algorithm is based on data collected from users. Verification study confirms a 97% success rate.

Acceptance Rate: 27%.

Nur Al-huda Hamdan, Ravi Kanth Kosuru, **Christian Corsten**, Jan Borchers. 2017. Run&Tap: Investigation of On-Body Tapping for Runners. *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (ISS '17). ACM, Brighton, United Kingdom, 280–286. ISBN: 978-1-4503-4691-7. DOI: 10.1145/3132272.3134140. http://doi.acm.org/10.1145/3132272.3134140.

Contribution and Benefits: Investigates on-body tapping as input for runners. Tapping data from motion-captured subjects on a treadmill shows that five targets per arm and two targets on the abdomen give 96% accuracy.

Acceptance Rate: 27%.

Christian Corsten, Bjoern Daehlmann, Simon Voelker, Jan Borchers. 2017. BackXPress: Using Back-of-Device Finger Pressure to Augment Touchscreen Input on Smartphones. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, Denver, Colorado, USA, 4654–4666. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025565. http://doi.acm.org/10.1145/3025453.3025565.

Contribution and Benefits: BackXPress lets users create force input with six fingers at the back of landscape-held smartphones to augment their frontal touchscreen interaction. We present design guidelines derived from three studies.

Acceptance Rate: 25%.

Christian Corsten, Andreas Link, Thorsten Karrer, Jan Borchers. 2016. Understanding Back-to-Front Pinching for Eyes-Free Mobile Touch Input. *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '16). ACM, Florence, Italy, 185–189. ISBN: 978-1-4503-4408-1. DOI: 10.1145/2935334.2935371. http://doi.acm.org/10.1145/2935334.2935371.

Contribution and Benefits: Investigates the influence of device thickness and tilt angle for back-to-front pinching on handheld touch devices. Pinch error significantly increases with thickness while the angle has no influence.

Acceptance Rate: 24%.

Christian Corsten, Christian Cherek, Thorsten Karrer, Jan Borchers. 2015. HaptiCase: Back-of-Device Tactile Landmarks for Eyes-Free Absolute Indirect Touch. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, Seoul, Republic of Korea, 2171–2180. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702277. http://doi.acm.org/10.1145/2702123.2702277.

Contribution and Benefits: Presents an interaction technique for eyes-free absolute indirect touch based on back-of-device tactile landmarks and proprioception. Improves tapping accuracy significantly (14% target size reduction) without software modification.

Acceptance Rate: 25%.

Simon Voelker, **Christian Corsten**, Nur Al-huda Hamdan, Kjell Ivar Øvergård, Jan Borchers. 2014. An Interaction Model for Grasp-Aware Tangibles on Interactive Surfaces. *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (ITS '14). ACM, Dresden, Germany, 279–282. ISBN: 978-1-4503-2587-5. DOI: 10.1145/2669485.2669494. http://doi.acm.org/10.1145/2669485.2669494.

Contribution and Benefits: Presents an interaction model for grasp-aware tangible objects used on capacitive interactive surfaces. Gives examples showing how the user benefits from this model that extends Buxton's Three-State Model for graphical input.

Acceptance Rate: 28%.

Christian Corsten, Ignacio Avellino, Max Möllers, Jan Borchers. 2013. Instant User Interfaces: Repurposing Everyday Objects as Input Devices. *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces* (ITS '13). ACM, St. Andrews, Scotland, United Kingdom, 71–80. ISBN: 978-1-4503-2271-3. DOI: 10.1145/2512349.2512799. http://doi.acm.org/10.1145/2512349.2512799.

Contribution and Benefits: Instant UIs allow everyday objects to be programmed as dedicated input devices by ad-hoc demonstration. Our Marker-free tracking system enables to assign semantic meaning to object poses and touches.

Acceptance Rate: 29%.

Book Chapters (Edited, archival)

Jan Borchers, **Christian Corsten**, Max Möllers, Simon Völker. 2013. Von der Fläche zur Kurve: Multi-Touch auf beliebigen Oberflächen und Objekten. *Multi-Touch: Interaktion durch Berührung*. Springer, Berlin, Heidelberg, 369–392. ISBN: 978-3-642-36113-5. DOI: 10.1007/978-3-642-36113-5_17. https://doi.org/10.1007/978-3-642-36113-5_17.

Contribution and Benefits: A chapter on multi-touch interaction techniques on flat and curved surfaces. Closes with touch and grasp interaction for physical objects and discusses how they can be repurposed as input devices.

Posters (Peer-reviewed, non-archival)

Nur Al-huda Hamdan, Marcel Lahaye, **Christian Corsten**, Jan Borchers. 2016. Presentation Strategies for Micro-Navigation in the Physical World. *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (CHI EA '16). ACM, San Jose, California, USA, 3062–3068. ISBN: 978-1-4503-4082-3. DOI: 10.1145/2851581.2892543. http://doi.acm.org/10.1145/2851581.2892543.

Contribution and Benefits: Provides insights into hierarchical micro-navigation in the physical world with relevance to AR systems. Study shows that in shallow hierarchies route aids and survey aids perform comparably in terms of navigation time and accuracy.

Acceptance Rate: 20%.

Simon Voelker, **Christian Corsten**, Nur Al-huda Hamdan, Kjell Ivar Øvergård, Jan Borchers. 2014. An Interaction Model for Touch-Aware Tangibles on Interactive Surfaces. *Proceedings of the 2014 CHI Conference on Human Factors in Computing Systems* (CHI EA '14). ACM, Toronto, Ontario, Canada, 1873–1878. ISBN: 978-1-4503-2474-8. DOI: 10.1145/2559206.2581273. http://doi.acm.org/10.1145/2559206.2581273.

Contribution and Benefits: Presents a work-in-progress interaction model for grasp-aware tangible objects. Extends Buxton's Three-State Model for graphical input.

Acceptance Rate: 31%.

Posters (Juried, non-archival)

Christian Corsten, Chat Wacharamanotham, Jan Borchers. 2013. Fillables: Everyday Vessels As Tangible Controllers with Adjustable Haptics. *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '13). ACM, Paris, France, 2129–2138. ISBN: 978-1-4503-1952-2. DOI: 10.1145/2468356.2468732. http://doi.acm.org/10.1145/2468356.2468732.

Contribution and Benefits: Tunes Tangible UIs ad-hoc by filling water into everyday objects. Reports how users can discriminate different filling levels that make virtual granularity (video navigation, virtual brush size) perceptible eyes-free.

Acceptance Rate: 29%. — (alt.CHI)

Christian Corsten. 2010. DragonFly: Spatial Navigation for Lecture Videos. *CHI '10 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '10). ACM, Atlanta, Georgia, USA, 4387–4392. ISBN: 978-1-60558-930-5. DOI: 10.1145/1753846.1754158. http://doi.acm.org/10.1145/1753846.1754158.

Contribution and Benefits: Presents DragonFly, an application for reviewing lecture recordings of mind map-structured presentations. Shows that DragonFly users are 50% faster in finding a specific scene in the lecture recording compared to using classic linear video control.

Acceptance Rate: 26%. — (CHI Student Research Competition)

Theses

Christian Corsten. 2012. *Co-Optjects: Instant User Interfaces Through Everyday Objects*. Master's Thesis. RWTH Aachen University, (January 2012). https://hci.rwth-aachen.de/publications/corsten2012.pdf.

Contribution and Benefits: Presents a series of studies and implementation of prototypes that enable users to repurpose physical everyday objects as input devices. Proof-of-concept study demonstrates how everyday objects can be used to control a TV and desktop video navigation.

★ *Friedrich-Wilhelm Award*

Christian Corsten. 2009. *DragonFly: Reviewing Lecture Recordings with Spatial Navigation*. Bachelor's Thesis. RWTH Aachen University, (October 2009). https://hci.rwth-aachen.de/publications/corsten2009a.pdf.

Contribution and Benefits: Presents the design and implementation of DragonFly, an application for reviewing lecture recordings of mind map-structured presentations. User study proves that DragonFly users are 50% faster in navigating video scenes compared to using a linear video control.

Typeset May 5, 2020