Text and Numerical Input on Mobile and Wearable Devices

by

Yuan Ren

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering & Computer Science

in the

Graduate Division

of the

University of California, Merced

Committee in charge:

Assistant Professor Ahmed Sabbir Arif, Chair
Professor Shawn Newsam
Associate Professor David Noelle

Spring 2024

The dissertation of Yuan Ren, titled "**Text and Numerical Input on Mobile and Wearable Devices**", is approved:

Chair	______________________	Date	__________
	______________________	Date	__________
	______________________	Date	__________

University of California, Merced

Text and Numerical Input on Mobile and Wearable Devices

Inclusive Interaction Lab
https://www.theiilab.com

Abstract

Text and Numerical Input on Mobile and Wearable Devices

by

Yuan Ren

Doctor of Philosophy in Electrical Engineering & Computer Science

University of California, Merced

Assistant Professor Ahmed Sabbir Arif, Chair

Text and numeric input and editing are prevalent in our daily digital interactions. Although text entry has been meticulously investigated by the human-computer interaction (HCI) community, numeric input and editing have not been as thoroughly examined. Consequently, the methods employed on mobile and wearable devices, such as smartwatches, are inefficient compared to their desktop counterparts. This inefficiency is primarily due to the smaller screen real estate of these devices, which cannot display all relevant information, the absence of haptic feedback provided by physical keyboards and keypads, and an excessive reliance on precise target selection, which is difficult on smaller screens. This dissertation investigates, designs, and evaluates novel methods for text and numeric input and editing for mobile and wearable devices by leveraging the existing capabilities of smartphones and smartwatches. The first part of this dissertation proposes three new numeric input and editing methods for smartwatches based on gestures, tilting, and force. The proposed methods enable users to actively switch between slow-and-steady and fast-and-continuous increments and decrements of numeric values during the input process. Results reveal that the gesture-based method yields a significantly faster input rate and is perceived as faster, more accurate, and the least mentally and physically demanding compared to the other methods. The second part explores the possibility of using a force-based approach to target selection on smartwatches and its use in character entry. The initial study identifies the most comfortable range of force on smartwatches. The subsequent study compares the performance of tap and force-tap in a Fitts' Law setting. Results revealed that force-tap is significantly better at selecting smaller targets, while tap outperforms force-tap for larger targets. We then develop two new force-based keyboards to demonstrate the feasibility of force input in practical scenarios. These single-row alphabetical keyboards enable character-level text entry by performing slides and varying contact force. In a user study, these keyboards yield about 4 wpm with about a 2% error rate, demonstrating the viability of force input on smaller screens. The third part of the dissertation presents GeShort, a novel method for one-handed text editing and formatting on mobile devices. It uses simple rules to facilitate direct cursor positioning, gestural shortcuts inspired by keyboard hotkeys for editing and formatting, and a floating clipboard to enable delayed, repeated, and block editing. A comparison between GeShort and the default Google keyboard reveals that users perform editing and formatting tasks about 11% and 22% faster, respectively, with GeShort, achieved by significantly reducing selection time by 11% and action time by 17%. A second study comparing the clipboard features of the two methods revealed that users perform advanced editing tasks 34% faster with GeShort. The findings presented in this dissertation illustrate that numeric and character entry and editing can be significantly improved in terms of speed, accuracy, and user preference. This is achieved by employing careful interface and interaction design, utilizing simple rules, and applying language models, all while making use of the existing capabilities of mobile and wearable devices. Importantly, this enhancement does not require the addition of extra sensors, new hardware, or the computational power necessary for complex learning models.

Publications

During my Ph.D. journey, I have published seven refereed articles, listed below. This dissertation includes numbers 3, 4, and 7. In addtion, I have one paper currently under preparation.

1. Wendy Haw, Yuan Ren, Kianna Ng, Ahmed Sabbir Arif. **2024**. Investigating the Effects of Self-selected Pleasant Scents on Text Composition and Transcription Performance. In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (**CHI 2024**). ACM, New York, NY, USA.

2. Veena Sumedh, Peter Ly, Yuan Ren, Cristina Sylla, Abigail Plata, Ahmed Sabbir Arif. **2024**. Impact of Static and Animated eBook Illustrations on Children's Engagement, Enjoyment, and Information Recall. In CHI Conference on Human Factors in Computing Systems Extended Abstracts (**CHI EA 2024**). ACM, New York, NY, USA.

3. Gulnar Rakhmetulla, Yuan Ren, Ahmed Sabbir Arif. **2023**. GeShort: One-Handed Mobile Text Editing and Formatting with Gestural Shortcuts and a Floating Clipboard. Proc. ACM Hum.-Comput. Interact. 7, **MHCI**, Article 212 (Sep. 2023), 23 pages.

4. Yuan Ren, Ahmed Sabbir Arif. **2023**. Investigating a Force-Based Selection Method for Smartwatches in a 1D Fitts' Law Study and Two New Character-Level Keyboards. In Proceedings of the 17th International Conference on Tangible, Embedded, and Embodied Interaction (**TEI 2023**). ACM, New York, NY, USA, 10 pages.

5. Ghazal Zand, Yuan Ren, Ahmed Sabbir Arif. **2022**. TiltWalker: Operating a Telepresence Robot with One-Hand by Tilt Controls on a Smartphone. Proc. ACM Hum.-Comput. Interact. 6, **ISS**, Article 572 (Dec. 2022), 26 pages.

6. Tafadzwa Joseph Dube, Yuan Ren, Hannah Limerick, I. Scott MacKenzie, Ahmed Sabbir Arif. **2022**. Push, Tap, Dwell, and Pinch: Evaluation of Four Mid-Air Selection Methods Augmented with Ultrasonic Haptic Feedback. Proc. ACM Hum.-Comput. Interact. 6, **ISS**, Article 565 (Dec. 2022), 19 pages. **Best Paper Award**.

7. Yuan Ren, Ahmed Sabbir Arif. **2021**. Stepper, Swipe, Tilt, Force: Comparative Evaluation of Four Number Pickers for Smartwatches. Proc. ACM Hum.-Comput. Interact. 5, **ISS**, Article 500 (Nov. 2021), 21 pages. **Honorable Mention Award**.

Awards

I have had the privilege of receiving several awards during my Ph.D. studies, which are listed below.

- UC Merced EECS Bobcat Travel Fellowship 2024
- UC Merced EECS Bobcat Travel Fellowship 2022
- UC Merced EECS Bobcat Summer Fellowship 2020

To my beloved daughter Chloe, you are a treasure beyond compare.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincere gratitude to the following individuals without whom this dissertation would not have been possible.

First and foremost, I am deeply indebted to my advisor, Professor Ahmed Sabbir Arif, for his unwavering support, invaluable guidance, and constant encouragement throughout this research endeavor. His expertise, patience, and dedication have been instrumental in shaping this work and shaping me as a scholar.

I extend my heartfelt appreciation to my dissertation committee members, Professor Shawn Newsam and Professor David Noelle, for their insightful feedback, constructive criticism, and scholarly input. Their expertise and commitment to excellence have enriched this study immeasurably.

I am indebted to my beloved husband, Ziqi Liu, whose unwavering belief in my capabilities and constant support have been the cornerstone that sustained me through the challenges of academia. His presence, marked by empathy, patience, and devotion, has been an immeasurable blessing.

My profound appreciation also extends to my parents and parents-in-law for their invaluable support and assistance with childcare. Their selflessness and readiness to extend a helping hand have granted me the precious time and space needed to concentrate on my studies and pursue my academic aspirations. I am truly fortunate to be enveloped by such a nurturing family network, and I am deeply moved by their love, encouragement, and unwavering confidence in my abilities.

I express my deepest gratitude to my daughter Chloe for gracing this world with her presence. Her arrival has brought huge joy and meaning to my life, and I am endlessly thankful for the gift of her existence.

To all my friends who have stood by me through thick and thin, their encouragement, positivity, and camaraderie have provided me with the strength and motivation to persevere, even during the most challenging times.

Finally, I would like to express my appreciation to University of California Merced for providing me with the resources, facilities, and opportunities essential for the completion of this dissertation.

This dissertation is a testament to the collective efforts of all those mentioned above, and I am profoundly grateful for their contributions to this academic endeavor.

Chapter 1

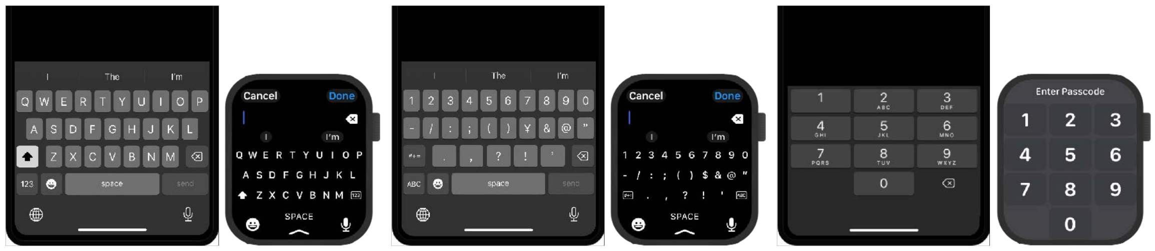
Introduction

In recent years, smartphones and smartwatches have evolved into staple consumer products [51, 91]. With their enhanced computational power, usability, and accessibility, these devices empower users to access information and input text and numbers without the need for a desktop computer. A recent survey of smartphone users revealed that approximately 75% use their devices for chatting and messaging, while nearly 70% engage in email communication [37]. Accessing online banking emerges as the third most popular activity among global respondents, closely following the aforementioned uses. Importantly, these activities necessitate the entering and editing of text and numbers. Another survey reported that smartwatch owners primarily use their devices for texts and notifications, activity tracking, and emails, with usage rates of 54%, 45%, and 25%, respectively [99]. Notably, approximately one-in-five U.S. adults (21%) regularly wear a smartwatch or a fitness tracker [133] to enter numerical data related to fitness activities, such as workout duration, distance, calories burned, or heart rate monitoring. Consequently, many of these activities also require the input and editing of text and numeric data[1]. Beyond these activities, text and numeric entry and editing also serve additional purposes, such as taking quick notes, entering passwords and access codes, making email address or phone number entries, scheduling appointments, or setting reminders. Therefore, the ability to effectively and effortlessly enter and edit characters[2] can enhance productivity by facilitating greater access to productivity tools and can significantly simplify our daily routines.

Although text entry has been extensively studied by the human-computer interaction (HCI) community, the area of numeric input and editing has not received as much attention. As a result, the methods used on mobile and wearable devices, such as smartwatches, are less efficient than those on desktop computers. This inefficiency is mainly due to the limited screen size of these devices, which restricts the display of all pertinent information, the lack of haptic feedback that physical keyboards and keypads offer, and a heavy dependence on precise target selection, which becomes challenging on smaller screens. Indeed, the primary method of text input for mobile and wearable devices is the virtual Qwerty keyboard (Fig. 1.1). This is a keyboard that replicates the layout of traditional physical keyboards on touchscreen devices, allowing for text input through the tapping of miniature virtual keys representing letters, numbers, and symbols. However, the Qwerty layout has its disadvantages, originating from its initial design intended to prevent jamming in mechanical typewriters,

[1]In this dissertation, we define "number editing" as instances in which an existing number is modified or a number is selected from a specified range.

[2]In this dissertation, the term "characters" encompasses all printable entities, including letters, numbers, and symbols.



(a) Virtual Keyboard (b) Virtual Keyboard with Numbers (c) Number Pad

Figure 1.1: Compact virtual Qwerty keyboards and numeric entry layouts on smartphones and smartwatches.

rather than to enhance typing speed or ergonomic comfort for users. Using this layout on smaller touchscreens often leads to slower typing speeds and increased finger strain due to the suboptimal key arrangement [14]. Additionally, as device screen sizes become smaller, ranging from laptops and tablets to smartphones and smartwatches, the keys themselves are reduced in size. This size reduction complicates selection and elevates error rates [10]. The Qwerty keyboard also facilitates the input and editing of numbers, but requires users to switch to a different layout to access a row of numbers at the top. In certain situations, such as when input is limited to numbers (e.g., entering a PIN code, phone number, or zip code), the keyboard interface is substituted with number pads and symbol entry layouts with different key arrangements. This necessitates learning multiple layouts and diverting focus from the primary task to switching between layouts. Thus, text and numeric entry and editing remain challenging on smaller touchscreen devices, like smartphones and smartwatches.

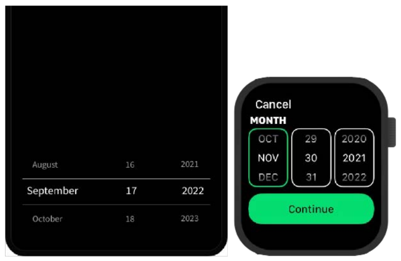
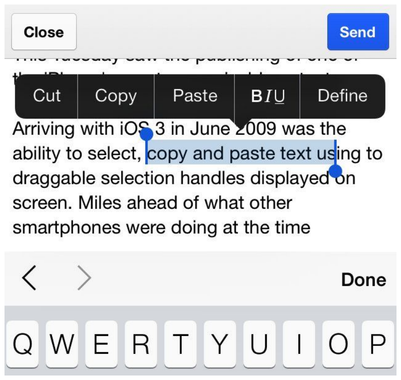


Figure 1.2: Number picker on a smartphone (left) and a smartwatch (right).

Number pickers offer an alternative method for entering numbers (Fig. 1.2). Unlike virtual keyboards that require users to input digits individually, number pickers present a default value and permit users to select a new number from a list or alter the default via a dual-segment control. Users can increase the value by swiping up or decrease it by swiping down, with continuous adjustment necessitating repeated swipes. However, number pickers fall short when dealing with large numeric ranges. Scrolling through long sequences to locate a specific value can be tedious, leading to prolonged selection times, increased fatigue, and potential frustration. Furthermore, on devices with small screens, such as smartwatches, the limited display area increases the challenge of navigating through vast number lists. Additionally, number pickers frequently require the use of the entire screen, thereby hindering or complicating the process of inline editing.



Arriving with iOS 3 in June 2009 was the ability to select, copy and paste text using to draggable selection handles displayed on screen. Miles ahead of what other smartphones were doing at the time

Figure 1.3: Cursor positioning, text selection, and editing features on an iOS-based device [53].

Like numbers, text enditing and formatting is also difficult on smaller screens. The process of text editing and formatting on mobile devices typically involve cursor positioning, activating "edit mode" via a long tap to display an edit toolbar and selection handles, adjusting the selection, and choosing a task from the toolbar (Fig. 1.3). For formatting tasks often hidden in a secondary menu, further navigation is required. The "fat-finger problem" complicates precise cursor placement, leading to errors and correction delays [132, 21]. The necessity for a dwell threshold and navigating multiple menus adds complexity and time, especially when using the device with one hand, a common scenario for mobile users under various circumstances [8]. Existing methods for mobile text editing lack support for efficient delayed, repeated, and batch editing, usually requiring a cumbersome sequence of actions for each task. The default Google keyboard, Gboard, has introduced a clipboard for some improvement in managing text snippets for later use, nevertheless accessing and utilizing this feature remains cumbersome and inefficient.

In summary, devices with smaller screens face the following challenges in character entry and editing.

- **Small key size**. Virtual Qwerty keyboards on these devices often have small keys, raising the likelihood of accidental taps or misinputs, a contrast to the more reliable physical keyboards.

- **Screen occlusion**. The issue of users' fingers obscuring the screen while typing on virtual keyboards limits visual feedback. This problem exacerbates as the target size decreases, leading to frequent errors and user frustration.

- **Lack of physical feedback**. The absence of tactile feedback on touchscreen devices increases user uncertainty during typing, as they lack the physical confirmation of their inputs. This absence necessitates constant visual monitoring to ensure accuracy.

- **Limited screen size**. The significantly smaller screens of these devices, especially smartwatches, limit the range of tasks that can be performed comfortably. Complex actions such as extensive swipes, multi-point panning, or intricate gestures are particularly challenging.

- **Reduced dexterity**. The necessity of using two hands for certain tasks adds to the difficulty, as the hand wearing the device cannot assist in input actions. This limitation forces users to operate the device solely with one hand, usually leading to decreased precision, slower input speeds, and an increased incidence of errors.

This necessitates the development of faster, more accurate, and more reliable techniques for character input and editing by leveraging the existing capabilities of smartphones and smartwatches as alternative input modalities.

1.1 Contributions

This work investigates, designs, and evaluates innovative methods for text and numeric input and editing on mobile and wearable devices, utilizing the existing capabilities of smartphones and smartwatches.

Initially, we address the challenge of number entry on smartwatches, where selecting small targets is problematic due to the fat-finger problem. Given that text entry on smartwatches represents an extreme case of target selection, requiring repetitive selection of miniature keys, text entry is particularly difficult. Traditional methods often involve multi-step processes for character input, further complicating text entry. Notably, number picking is a frequent input task on smartwatches. Existing devices like the Apple Watch and Android Wear use an input stepper for number picking, demanding users spin a wheel to input and edit numbers, which can be distracting and fatiguing. We propose three innovative methods for number picking on smartwatches: directional swipes, wrist twisting, and contact force variation. Our results indicate that the swipe-based method, in particular, offers a significantly faster input rate in both stationary and mobile contexts, and during both individual and inline number editing, being perceived as fast, accurate, and less demanding mentally and physically.

Next, we explore target selection on smartwatches, especially for small targets, using a force-based approach. After determining the most comfortable force range for smartwatch interaction, we conducted a 1D Fitts' law study comparing tap and force-tap performance. Our findings suggest that force-tap excels at selecting smaller targets, while tap is more efficient for larger targets. This research led to the development of two new force-based keyboards, showcasing the practicality of force inputs on smaller screens.

Lastly, we focus on text editing and formatting on mobile devices, a critical need given their widespread use. The tasks of precise cursor positioning and text selection are hindered by the fat-finger issue and screen occlusion, often leading to errors and requiring additional correction time. We propose a novel method for one-handed text editing and formatting on mobile devices, introducing gestural shortcuts. This system uses straightforward rules for cursor placement, gestural shortcuts based on keyboard hotkeys for editing and formatting, and a floating clipboard for complex editing tasks. Our research shows that this method significantly outperforms standard techniques, reducing both selection and action times.

1.2 Outline

The dissertation is organized as follows. Chapter 2 introduces three innovative methods for number input on smartwatches, achieved through directional swipes, wrist twisting, and varying screen contact force. These methods contrast with the standard number picker by offering users the ability to switch between slow-and-steady and fast-and-continuous adjustments. We assessed these methods through two user studies: the first comparing them to the conventional input stepper in both stationary and mobile settings, and the second evaluating their performance in handling individual numbers and numbers within textual contexts. Chapter 3 explores the use of a force-based approach for selecting targets on smartwatches. It establishes the most comfortable force range for users and conducts a 1D Fitts' law study to compare tap and force-tap performance. This chapter also details the design and testing of two new force-based text entry techniques, illustrating the feasibility of this approach. Chapter 4 details a novel strategy for one-handed text editing and formatting on mobile devices, employing straightforward rules for direct cursor positioning, gestural shortcuts modeled after keyboard hotkeys, and a floating clipboard for complex editing tasks. It presents a comparison between this method and the default Google keyboard across various editing and formatting activities. Chapter 5 concludes the dissertation, outlining future research prospects and possible expansions of the work presented.

Chapter 2

Number Entry & Editing on Smartwatches

2.1 Introduction

Smartwatches are becoming increasingly popular among mobile users. A recent survey reported that roughly one-in-five U.S. adults (21%) regularly wear a smartwatch or a fitness tracker [133]. Yet, interaction with these devices is mostly limited to receiving notifications on emails, text messages, and social media activities [33]. This is primarily due to the smaller screen sizes of smartwatches, which limits the types of actions users can perform on these devices. Text entry on these devices is particularly difficult as most existing text entry methods use miniature keys that are difficult to select, causing frequent errors [14]. Some methods also use multi-step approaches, where users have to perform a sequence of actions to input one character. Due to these challenges in text entry, number picking is arguably the most frequent input task performed on smartwatches. For illustrative

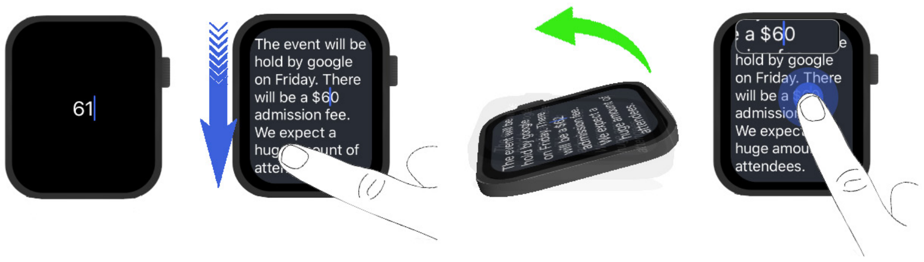

61
The event will be hold by google on Friday. There will be a $60 admission fee. We expect a huge amount of attendees.
a $60

Figure 2.1: The four number pickers investigated in this work, from left: input stepper (the default number picker on most smartwatches), swipe (in this figure, the user is swiping down to decrease a value), tilt (in this figure, the user is tilting up to increase a value), and force (in this figure, the user is variating contact force to change a value). Unlike the default input stepper, the new methods do not display numeric values in full-screen when selected, instead display a magnifier window to display the value changing (see the last image).

purposes, current Apple smartwatches come with 40 pre-installed apps[1], of which four partially support alphanumeric text and emoticon entry, eight support number entry using a number picker, and the remaining 28 are only for acquiring and viewing information.

Unlike traditional input methods, with which users enter one digit of a number at a time (such as a virtual keyboard), number pickers display a default value on the screen and enable users to pick a different number either from a list or by increasing or decreasing the default value using a two-segment control. The default number picker on most smartwatches is "input stepper". It uses a spinning number wheel metaphor, where flicking up on the screen spins the wheel to increase the value and spinning the wheel by a down flick decreases the value. Faster flicks spin the wheel faster and vice versa. Once flicked, the wheel keeps on spinning for some time, gradually slowing down to a full stop. Thus, repeated flicks are needed for continuous spinning of the wheel. Input stepper is also the default number picker on most smartphones. However, on smartphones, this method is used by a select number of apps (i.e., the clock) since the dominant method for entering numeric and other characters on these devices is virtual keyboards, while on smartwatches it is used almost exclusively to enter numbers. Section 2.3.1 describes this method. There are several limitations of this method. First, it does not usually enable editing inline values (numbers embedded in text, e.g., *"12:30"* in *"let's meet at 12:30 pm"*). Even when it does, it displays the spinner in full-screen, forcing users to take their attention away from the details on the screen that may include important details about the target number (e.g., available time slots for a meeting). Second, this method is impractical when the difference between the default and the intended values is very large (e.g., when changing *"$50 daily"* to *"$1,500 monthly"*) as it requires repetitive flicks, which can be physically and mentally challenging. Yet, to the best of our knowledge, no prior works studied number pickers on smartwatches or proposed alternatives to the default number picker.

In this paper, we develop three new number pickers in a rigorous design process. We then compare the performance of these methods with the default method in two empirical studies: one comparing their performance in stationary and mobile settings and another with individual and inline numeric values. The remainder of this chapter is organized as follows. First, we review the existing works in the area. We then describe the default number picker and introduce the proposed number pickers. We discuss the design considerations for the new methods. We then present the findings of two user studies comparing the four number pickers. Finally, we conclude by reflecting on future extensions of the work.

2.2 Related Work

Numerous works have explored swipe-based, tilt-based, and force-based input and interaction methods for larger touchscreen-based devices like interactive tabletops and walls [75, 108], tablets [135, 59], and smartphones [112, 127, 5, 39, 128, 36, 47, 124, 46]. However, these alternative interaction methods have not been well explored in the context of smartwatches. Interaction with smartwatches is fundamentally different from these devices as not only they are smaller in size but also have different holding and usage patterns. Unlike other touchscreen-based devices, smartwatches are worn on the wrist, which limits its interaction space as users cannot use the fingers of the watch-hand to interact with the device.

[1]Apps on Apple Watch, https://support.apple.com/en-gb/guide/watch/apdf1ebf8704/watchos

There has been an increased interest in text entry methods for smartwatches. Majority of these methods are miniature versions of the standard Qwerty layout that enable users to increase the size of the keys by either tapping or flicking on the screen [110, 41, 76]. There are also alternative miniature layouts that map the English alphabet to a fewer number of keys than Qwerty, then disambiguate the input using a language model [67, 85]. Two recent methods, WatchWriter [64] and SwipeRing [115], enable users to enter text by connecting the keys or the zones containing the letters of the intended word on the screen. Another method, WrisText [63], enables connecting the zones by whirling the wrist of the watch-hand in joystick-like motions. However, these methods use language models to disambiguate the input, and do not offer effective mechanisms for entering numeric values. Arif and Mazalek [14] provide a comprehensive review of the existing text entry methods for smartwatches.

Some have used smartwatches to control other systems. Duet enables controlling a smartphone using the spatial configuration of a smartwatch [42]. Users can tap and swipe on the smartwatch and perform wrist gestures to interact with a smartphone. Likewise, MultiFi lets users use extended widgets on a smartwatch in augmented/mixed reality, where users select a menu item by either tapping on the screen or pointing at the item. Some have also used smartwatches as active tangible in tangible-tabletop systems [65]. Actible [57], for example, augments a smartwatch with custom hardware to enable an expanded set of tangible interactions on interactive tabletops, including shaking, tilting, stacking, neighboring, and on-screen gestures [22, 50, 103]. These methods, however, were designed keeping interaction with other systems in mind, thus are not suitable for interacting with the smartwatches. Besides, the smartwatch-based active tangibles are usually used on a tabletop, not worn on the wrist. A different line of research exploits the accelerometer and gyroscope sensors of smartwatches [138, 139] or external sensors (e.g., infrared sensors [84, 87] and chest-mounted camera [98]) to track hand and finger movements. These works, however, are outside the scope of this research.

A few works have also explored tilt-based and force-based interaction methods on smartwatches.

To the best of our knowledge, no prior work developed or compared swipe-based, tilt-based, and force-based interaction methods on a smartwatch, especially in the context of number entry.

Table 2.1: Actions associated with the four number pickers.

Method	**Increment**	**Decrement**	**Acceleration Rate**	**Continuous Spinning**
Input Stepper	Flick up	Flick down	Pace of flick	Repetitive flicks
Swipe-Based	Swipe up	Swipe down	Length of swipe	Swipe and hold
Tilt-Based	Tilt the wrist up	Tilt the wrist down	Angle of tilt	Tilt and hold
Force-Based	Increase contact force	Decrease contact force	Level of force	Hold contact force

2.3 The Four Number Pickers

This section presents the default input stepper and the new swipe-based, tilt-based, and force-based number pickers. The design of the new methods were refined in iterative design steps. Table 2.1

summarizes the functionality of the four methods. In addition, these methods share the following behaviors.

- **Selection.** All methods enable selecting a numeric value for increment or decrement by tapping on it. When the selected value is embedded in text (inline values), the input stepper displays a full-screen virtual number wheel containing all legal values, which is the default behavior on most smartwatches. The swipe-based, tilt-based, and force-based methods, in contrast, do not display the number wheel in full-screen, instead change the value directly in the text. For conjoint values (i.e., multiple numbers connected with infixes), the input stepper displays one number wheel per segment. For example, for the time value "12:30:44", it displays three wheels, one for each segment, which is the default behavior on most smartwatches. With the new methods, however, users individually tap on the three parts of the value to change the respective parts.
- **Continuous Spinning.** The new methods enable continuous spinning of the number wheel with a single action for faster increment and decrement when the difference between the current and the intended value is large (e.g., changing "110" to "250"). The default input stepper does not provide the support for this, rather requires users to repeatedly flick on the screen for continuous spinning of the wheel. This feature is further discussed in Sections 2.3.1.
- **Auditory Feedback.** All four methods provide auditory feedback on spinning the number wheel (a spinning wheel sound, like the default Apple iOS input stepper).
- **Visual Feedback.** Since the new methods do not switch to a full-screen mode when users select a numeric value embedded in text, they provide additional visual feedback to assure that users can see the value changing when their finger occludes the embedded value. Particularly, these methods display a magnifier window further from the finger (Fig. 2.1).

For the design and development, we simulated an Apple Watch 5 on an Apple iPhone X, where only the smartwatch display was active (Section 2.4.1 provides further details). We could not use an actual smartwatch since current smartwatches do not provide the support for continuous force detection. Apple Watch 5 includes a force sensor that can only *"distinguish between a light tap and a deep press"* [55]. We optimized all methods for the simulated smartwatch for a fair comparison between them. It is relatively common to use smartphones or tablets to study interactions with smartwatches due to technological limitations of current smartwatches (e.g., [41, 110, 94]).

2.3.1 Input Stepper

Input stepper is the default number picker on most smartwatches (Fig. 2.2). However, different mobile operating systems use different names to refer to this method, such as *Spinner* and *Spinning wheel*. When users tap on a numeric value, it displays a full-screen virtual number wheel containing all legal values. Spinning the wheel by flicking up on the screen increases the value and spinning the wheel by flicking down decreases the value[2]. The pace of the increment and decrement is determined based on how fast users flick on the screen. Faster flicks rotate the wheel faster and slower

[2]Some smartwatches use a reversed mapping, where spinning the wheel by flicking up decreases the value and spinning the wheel by flicking down increases the value. We decided against using this mapping to maintain interaction consistency between the four examined number pickers.

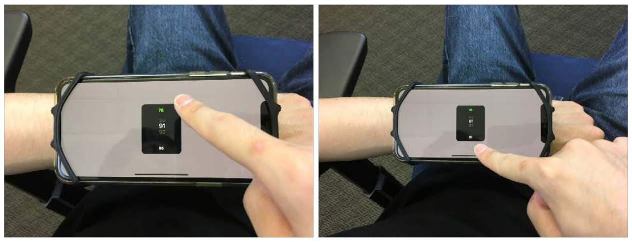
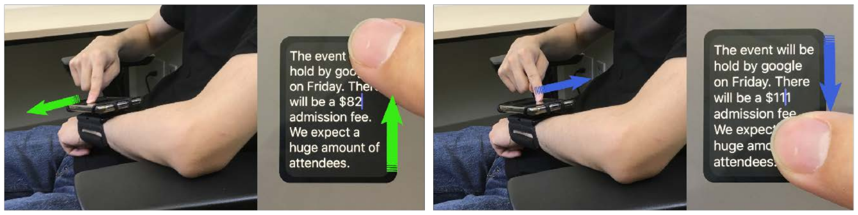


Figure 2.2: Input stepper is the default number picker on most smartwatches: it displays a full-screen number wheel that users spin by flicking up or down to increment or decrement a numeric value, respectively.

flicks rotate the wheel slower. When the desired number is displayed, users pick it by tapping on it. Continuous spinning is not fully supported by the default method. Once flicked, the wheel keeps on spinning for some time, gradually slowing down to a full stop. Hence, repeated flicks are needed for continuous spinning of the wheel. We implemented this method using the default iOS SDK control library[3], without any customization.



The event
hold by goo
on Friday. The
will be a $82
admission fee.
We expect a
huge amount of
attendees.

The event will be
hold by google
on Friday. There
will be a $111
admission fee
We expect
huge amo
attendees.

Figure 2.3: Swipe-based number picker: the user swipes up to increment and swipes down to decrement a numeric value.

2.3.2 Swipe-Based Picker

The swipe-based picker enables users to pick numbers by performing swipes (Fig. 2.3). Tapping on a numeric value activates an invisible number wheel containing all legal values. Users then swipe up or down on the screen to increment or decrement the value, respectively. One difference between the default input stepper and swipe is: with the former, users perform a short quick flick, while with the latter, users perform a steady swipe. A single swipe changes the value by one unit. Users can activate continuous spinning of the number wheel by stroking and holding the swipe for 850 milliseconds. Lifting the finger deactivates continuous spinning. The pace of a spin is determined based on the length of the swipe. Holding a *long* swipe increases or decreases the value faster, while holding a *short* swipe increases or decreases the value slowly. This enables users to actively pick a change rate appropriate for the task. Based on the findings of a pilot study (N = 12, M = 26.0 years) investigating various custom and commonly used functions [35, 44] for mapping control

[3]Apple Developer, Pickers - Controls - iOS - Human Interface Guidelines, https://developer.apple.com/design/human-interface-guidelines/ios/controls/pickers.

movements to the movements of a display object, referred to as the control-display (CD) gain, we used the following function to determine the pace of a spin.

$$t(l) = \frac{1}{2} \times \left(1 - \frac{l}{h}\right)^3 \tag{2.1}$$

Where $t(l)$ is the pace of a spin relative to the length of a swipe l in pixels, and h is the height of the active smartwatch touchscreen in pixels. Note that we also considered a temporal approach to determine the pace of a spin, where the pace increases in proportion to the duration of a swipe-hold (the wheel keeps on spinning faster as users continue with holding a swipe). However, it performed poorly compared to the proposed approach in terms of speed and accuracy in another pilot study (*N* = 6, M = 26.0 years). Besides, most participants found the mapping confusing.

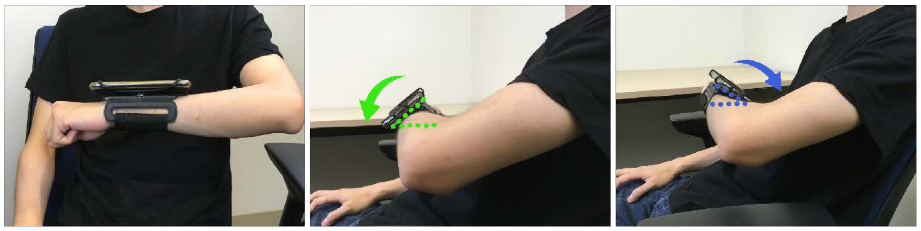


Figure 2.4: Tilt-based number picker: the user tilts the device away from the body (up) or towards the body (up) to increment and decrement a numeric value, respectively.

2.3.3 Tilt-Based Picker

The tilt-based method enables users to pick numbers by tilting their smartwatch towards or away from their body, interpreted as tilting down and up, respectively (Fig. 2.4). Tilting the device once then returning to the initial position changes the value by one unit. For continuous spinning of the invisible number wheel, users tilt and hold the position for 850 milliseconds. Bringing the device back to its initial position deactivates continuous spinning. The pace of a spin is determined based on the angle of the tilt. Tilting the device at a steeper angle increases or decreases the value faster, while tilting it at a slight angle increases or decreases the value slowly. This enables users to actively pick a change rate appropriate for the task. First, we conducted a pilot study (*N* = 6, M = 26.0 years) to explore various custom and commonly used functions for CD gain [35, 44] to map different tilt angles to different spin pace. Selecting the most effective function was particularly challenging since tilting the device too much not only caused irritation and fatigue but also made the text illegible. Hence, based on Dunlop et al's [54] recommendation, we tested $15° \pm 9$ angle as a possible range for tilt. However, it was proven to be too sensitive for precise pace section in a second pilot study (*N* = 12, M = 27.08 years). Besides, we observed that tilting the device towards the body is more difficult than tilting away from the body, thus using the same range for the up and down tilts is impractical. Based on this, we changed the range and selected the following function to determine the pace of a spin based on the findings of a third pilot study (*N* = 12, M = 26.01 years).

$$t(a) = \begin{cases} \frac{3}{5} \times \left(1 - \frac{|a - up\_optimal|}{up\_threshold}\right)^{3.2} & \textit{when tilting up} \\ \frac{3}{5} \times \left(1 - \frac{|a - down\_optimal|}{down\_threshold}\right)^{3} & \textit{when tilting down} \end{cases} \tag{2.2}$$

Where $t(a)$ is the pace of a spin relative to the tilt angle a, $up\_threshold$ and $down\_threshold$ are the maximum possible up and down tilts $[-40°, 30°]$ based on the anatomy of the human wrist [97], and $up\_optimal$ and $down\_optimal$ are the optimal up and down tilts $[-30°, 20°]$ based on the findings of the pilot studies. Like the swipe-based method, we also tested a temporal approach to determine the pace of a spin, where holding a tilted position for longer increased the pace of the spin and vice versa. However, it was significantly slower, more error prone, and caused irritation and fatigue in the pilot study.

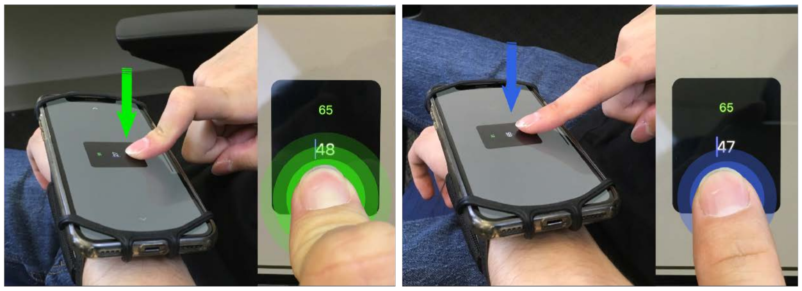


Figure 2.5: Force-based number picker: the user applies extra force to increment and soft force to decrement a numeric value.

2.3.4 Force-Based Picker

The force-based method enables users to pick numbers by variating contact force on the screen. Increasing contact force increases the value and decreasing contact force decreases the value (Fig. 2.5). The default Apple iOS SDK returns a value between 0 and 6.67 for the amount of force imparted by the user's finger onto the screen. We normalized it to the interval from 0 to 1 by dividing the received force value by the maximum force (6.67). Then, based on the findings of a pilot study (*N* = 15, M = 27.7 years), we segmented this range into three force levels: soft (from 0 to 0.15), regular (from 0.15 to 0.3), and hard (from 0.3 to 1). Initially, we attempted an adaptive approach that adjusted the levels of force based on users' typical contact force. However, this method failed to accurately predict the three levels of force in a pilot study investigating different positions and posture, including when standing, sitting, and walking (*N* = 15, M = 23.9 years). It also bothered the participants that they could not actively control the force levels. Further, it required a substantial amount of training data for each participant, which can be challenging and often impractical in real-world scenarios. The fixed range approach used in this work, in contrast, performed well in prior studies with smartphones [16, 19], as well as in a third pilot study (*N* = 13, M = 28.46 years). Participants were able to learn the three levels and replicate those in various settings without any major difficulties.

With the force-based method, changing the level of force once then returning to the regular force changes the value by one unit. For continuous spinning of the invisible number wheel, users change the level of force and hold that level for 850 milliseconds. Changing the level or lifting the finger deactivates continuous spinning. This method uses a temporal approach to determine the pace of a spin in continuous spinning. Holding a specific level of force longer increases or decreases the value faster at the following rates: 1 digit per 600 milliseconds for 2 seconds, 1 digit per 250 milliseconds for 5 seconds, and 1 digit per 5 milliseconds. This incremental pace rate was selected based on the findings of a fourth pilot study (N = 6, M = 26.5 years), where users could alter the contact force to restart the rate, enabling them to actively pick a rate appropriate for a task. To keep users informed, this method provides haptic feedback on each force level change. That is, the device vibrates for 200 milliseconds, as recommended by Kaaresoja and Linjama [81]. Since users do not necessarily have to lift their finger to select a value with this method, once selected, they could slide their fingers to a different value to edit it.

2.4 Experiment 1: Seated Vs. Walking

We conducted a user study to compare the four number pickers in both stationary and mobile settings. The study protocol was reviewed and approved by the Institutional Review Board (IRB). The study was completed before the World Health Organization (WHO) declared the outbreak of COVID-19 a pandemic.

2.4.1 Apparatus

We used an iPhone X (43.6×70.9×7.7 mm, 174 grams) running on iOS version 12.1 at 1125×2436 pixels resolution in the user study. We developed a custom app using the default iOS SDK to simulate an Apple Watch 5's 740 mm^2 display area (312×390 pixels) on the smartphone. We made the surrounding area of the simulated smartwatch touch-insensitive to avoid the effects of accidental touches during the study. We used a smartphone instead of an actual smartwatch since existing smartwatches cannot detect the exact level of force applied on the screen (see Section 2.3). Apple Watch detects only the absence and presence of extra force. We used a wristband with silicone phone holder (55.5 grams) to attach the smartphone to the wrist of the participants like a smartwatch (Figure 2.6). The wristband held the device on the wrist firmly, thus participants did not have to hold it steady with the fingers of the other hand, although we noticed a few participants occasionally doing that. The holder was 180° rotatable but we did not enable participants to rotate the device during the study to eliminate a potential confound. We used a Fitness Reality TRE5000 electric treadmill to simulate walking, which is common practice in controlled studies (e.g., [24, 26, 104, 129]).

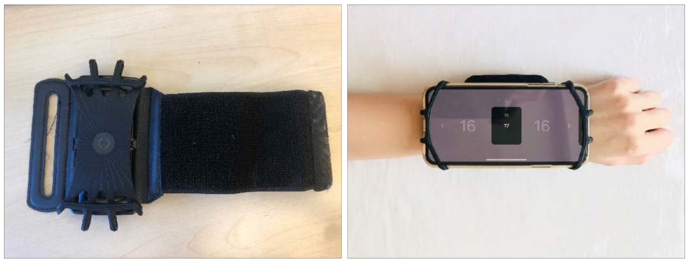


Figure 2.6: The wristband and the device used in the user study.

2.4.2 Participants

Twelve participants voluntarily took part in the study. None of them participated in the pilot studies. Their age ranged from 20 to 39 years (M = 25.27, SD = 5.5). Six of them identified themselves as female and six identified as male. Ten of them were right-handed and two were left-handed. All right-handed participants chose to wear the device on their left hand and interact with the device using the right hand, while all left-handed participants chose to wear it on the right hand and use the other hand for interaction. They all were experienced smartphone users (M = 8.36 years of experience, SD = 1.9). Two of them owned a smartwatch (M = 3.0 years of ownership, SD = 2.8). Four of them also had experience using force as an input modality on Apple iPhone devices (M = 4.6 years of experience, SD = 3.6). They all received U.S. $10 for participating in the study.

2.4.3 Design

We used a within-subjects design for the study, where the independent variables were *setting* and *method*, and the dependent variables were the following performance metrics.

- **Task completion time** (seconds) is the average time it took to change one presented value to the target value.
- **Actions per task** is the average number of actions, including taps, swipes, tilts, and different levels of force, performed to change one presented value to the target value.

Besides, participants were asked to complete the following questionnaires.

- A usability questionnaire that asked participants to rate various aspects of the examined number pickers on a 7-point Likert scale. It included four questions from the SUS questionnaire [31] and two custom questions. The four questions from SUS were about the willingness to use when seated and walking (SUS Q#1), ease of use (SUS Q#3), and learnability of the methods (SUS Q#7). The two custom questions were about the perceived speed and accuracy of the methods. These questions were used since SUS does not include questions about system speed or accuracy.
- A perceived workload questionnaire that included three questions from the NASA-TLX questionnaire [107] about mental demand (NASA-TLX Q#1), physical demand (NASA-TLX Q#6), and frustration (NASA-TLX Q#6).

We did not use the full SUS and NASA-TLX questionnaires to reduce the time and effort needed in the study. These questionnaires include 16 questions in total, which would have resulted in ($2 \times 4 \times 16 =$) 128 questions in the study. Hence, we only used the questions that are most relevant to our investigation. Hart [72], the creator of NASA-TLX, identified using a subset of the questions as one of the most common usage of the questionnaire and did not discourage it. Leaving some questions out of the SUS questionnaire is also common [96]. Note that we evaluated each scale individually rather than calculating a single score per questionnaire to eliminate the possibility of biases in factor analyses.

Participants were asked to complete both questionnaires upon the completion of the study to increase the reliability of the data as it enabled them to compare the efforts needed with each method while rating them. We acknowledge that this increases the chance of the *context effect*, which suggests participants tend to rate the latter methods more demanding than the ones they did earlier. However, Hart [71] found it to be *"typical of subjective ratings in general"* and argued that it can be avoided by being *"careful to control context effects"*, which we did by counterbalancing the conditions using a balanced Latin square. This assured that participants experienced the methods in different orders. Prior studies showed it to be an effective approach to mitigate context or order effect [101, pp. 177–181]. In summary, the design was as follows.

12 participants $\times$

2 settings (seated and walking, *counterbalanced*) $\times$

4 methods (the default stepper, swipe, tilt, and force, *counterbalanced*) $\times$

15 random two-digit values between 9 and 100

= 1,440 numeric values in total, excluding practices.

2.4.4 Experimental Tasks

During the study, the app presented one numeric value at a time and asked the participants to change it to a target value using the method under investigation (Fig. 2.7). Both the presented and the target values were two-digit numbers between 9 and 100. All tasks were randomly generated for each participant, making sure that there were equal number of increment and decrement tasks, and each task had its equal counterpart. That is, the system paired each increment task with an equivalent decrement task and vice versa, where the presented and target values had the same edit distance. For example, for a decrement task: change "68" to "43", there was an equivalent increment task: change "23" to "48" ($68 - 43 = 48 - 23 = 25$). We used numbers between 9 and 100 based on our observation that two-digit numbers, e.g., time (h:m:s), volume (0–100%), etc., are the most commonly edited numeric values on smartwatches[4]. Further, it allowed us to constrain each study session within one hour, reducing any potential effects of fatigue. It also enabled us to evaluate the new methods with numeric values with which the default input stepper is the most effective. The default input stepper method is likely to take more time and effort to edit larger numbers (> 100) as it would require repeatedly flicking on the screen to keep the wheel spinning. While with the proposed methods, users can continue spinning the wheel without performing additional actions until the target number is reached.

2.4.5 Walking Speed and Safety

The treadmill was set on 1.0 mph ($\sim$1.6 km/h) during the walking condition. We selected this rate based on a prior study that showed that users usually maintain a walking speed between 0.9 and 1.2 mph (1.5 and 2.0 km/h) when using a mobile phone [105]. Appropriate safety measures were taken during this condition. All participants were asked to attach the treadmill safety key to their clothing and wear a bike helmet to prevent injuries in case of an unexpected slip, trip, or fall. Besides, there were mandatory breaks between the conditions to prevent exhaustion for using the treadmill.

[4]Besides, informal investigations suggest that single- and two-digit numbers are the most frequently used [52] and the most popular [25].

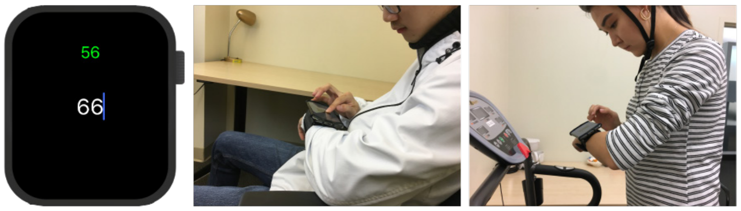


Figure 2.7: A decrement task (change "66" to "56") displayed on the study app (left), and two participants taking part in the study while seated and while walking on a treadmill, respectively (right).

2.4.6 Procedure

The study was conducted in a quiet room, one participant at a time. Upon arrival, we explained the research to all participants and collected their consents. They then completed a demographics and mobile usage questionnaire. The main study started after that. First, we demonstrated the first method and enabled them to practice with it by performing two increment and two decrement tasks. They were then asked to perform the experimental tasks (Section 2.4.4) both when seated and when walking in a counterbalanced order. They were instructed to perform the tasks as fast as possible. After successfully completing a task, they tapped on a button outside the smartwatch area to see the next task. Once done with all tasks, we demonstrated the second method, enabled them to practice with it, and asked them to perform the experimental tasks. This process continued until they experienced all methods. The methods were also introduced in a counterbalanced order. Upon completion of the study, the participants completed a questionnaire where they rated various aspects of the four methods on a 7-point Likert scale, and the perceived mental and physical demands and frustration using the NASA-TLX [107] questionnaire.

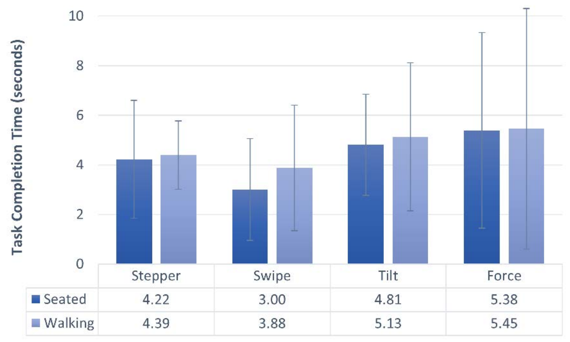
2.4.7 Results

A Shapiro-Wilk test revealed that the response variable residuals are normally distributed. A Mauchly's test indicated that the variances of populations are equal. Thus, we used a repeated-measures ANOVA to analyze the quantitative data. In contrast, we used a Friedman test to analyze the questionnaire data. We also report effect sizes of all statistically significant results: eta-squared (η^2) for ANOVA and Kendall's W for Friedman test [11]. Eta-squared uses Cohen's interpretation [45], where 0.01 constitutes a small, 0.06 constitutes a medium, and over 0.14 constitutes a large effect. Kendall's W uses a different interpretation by Cohen [45], where 0.1 constitutes a small, 0.3 constitutes a medium, and over 0.5 constitutes a large effect. There were no significant effects of the order of conditions on the dependent variables ($p > .8$), which suggests that counterbalancing worked [101, pp. 177–180].

2.4.7.1 Task Completion Time

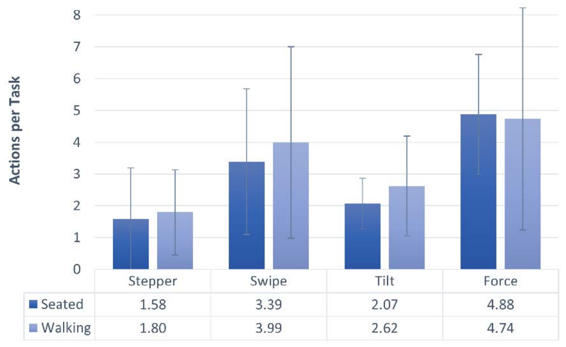
An ANOVA identified a significant effect of method on task completion time ($F_{3,33} = 23.76, p < .000005, \eta^2 = .07$)[5]. An ANOVA failed to identify a significant effect of setting ($F_{1,11} = 3.14, p =$

[5]The p value was too small for the NCSS and SPSS statistical software to display the exact value as they and most other statistical software display values up to six decimal places.



	Stepper	Swipe	Tilt	Force
■ Seated	4.22	3.00	4.81	5.38
■ Walking	4.39	3.88	5.13	5.45

(a) Task completion time (seconds)



	Stepper	Swipe	Tilt	Force
■ Seated	1.58	3.39	2.07	4.88
■ Walking	1.80	3.99	2.62	4.74

(b) Actions per task

Figure 2.8: Average task completion time and actions per task for all methods in the two settings (stationary and mobile). Error bars represent ± 1 standard deviation (SD).

.10). There was also no significant method $\times$ setting interaction effect ($F_{3,33} = 1.17, p = .33$). A Tukey-Kramer Multiple-Comparison test revealed that swipe was significantly faster than all other methods. Figure 2.8a illustrates average task completion time for the four methods in both settings.

2.4.7.2 Actions per Task

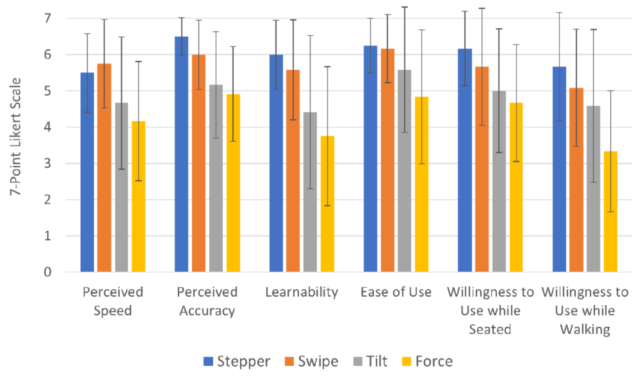
An ANOVA identified a significant effect of method on actions per task ($F_{3,33} = 14.71, p = .000003, \eta^2 = .18$). An ANOVA also identified a significant effect of setting ($F_{1,11} = 8.02, p = .016, \eta^2 = .003$). However, there was no significant method $\times$ setting interaction effect ($F_{3,33} = 1.28, p = .30$). A Tukey-Kramer Multiple-Comparison test revealed that stepper and tilt required significantly fewer actions per task compared to swipe and force. Figure 2.8b illustrates average actions per task for the four methods in both settings.

2.4.8 User Feedback

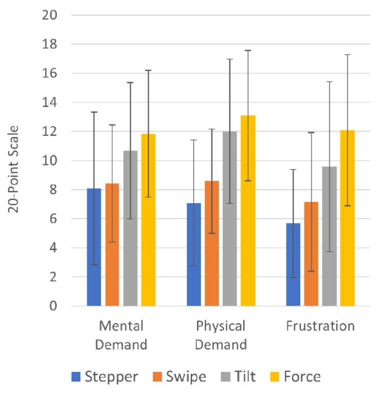
We used a non-parametric Friedman test to analyze the questionnaire data. Here, we present raw TLX scores by analyzing the sub-scales individually, which is a common modification made to NASA-TLX [71]. Note that the evidence is inconclusive about whether raw TLX is more sensitive, less sensitive, or equally sensitive compared to the original version [71].

2.4.8.1 Perceived Performance and Preference

A Friedman test identified a significant effect of method on perceived speed ($\chi^2 = 9.31, df = 3, p = .025, W = .26$), perceived accuracy ($\chi^2 = 20.40, df = 3, p = .0001, W = .57$), learnability ($\chi^2 = 15.61, df = 3, p = .001, W = .43$), ease of use ($\chi^2 = 18.47, df = 3, p = .0003, W = .51$), and willingness to use in both stationary ($\chi^2 = 10.07, df = 3, p = .01, W = .28$) and mobile settings ($\chi^2 = 12.49, df = 3, p < .006, W = .35$). Participants rated picker and swipe substantially higher in all aspects compared to tilt and force. Force received the poorest ratings of all methods. Figure 2.9a illustrates average user ratings of the four methods.



(a) Usability questionnaire



(b) Partial NASA-TLX questionnaire

Figure 2.9: Average user ratings of the four methods on a 7-point Likert scale, where "1" to "7" represented "strongly disagree" to "strongly agree" (left) and on NASA-TLX's 20-point scale, where "1" to "20" represented "no demand" to "extreme demand" (right). Error bars represent ± 1 standard deviation (SD).

2.4.8.2 Mental and Physical Demand

A Friedman test identified a significant effect of method on mental demand ($\chi^2 = 11.65, df = 3, p = .009, W = .32$), physical demand ($\chi^2 = 17.88, df = 3, p = .0004, W = .5$), and frustration ($\chi^2 = 19.41, df = 3, p = .0002, W = .54$). Participants found picker and swipe to be the least mentally and physically demanding compared to tilt and force. They also found the former methods to be the least frustrating. Force was rated substantially higher in terms of mental and physical demand, as well as frustration, compared to the other methods. Figure 2.9b illustrates average user ratings of the four methods.

2.5 Discussion

Results revealed that swipe was the fastest of all methods in both stationary and mobile settings. The average task completion time for stepper, swipe, tilt, and force were 4.31 (SD = 1.8), 3.44 (SD = 3.7), 4.97 (SD = 2.6), and 5.42 (SD = 3.7) seconds, respectively. On average, swipe was about 30–45% faster than the other methods, regardless of the fact that it required significantly more actions per task than picker and tilt. This is likely because participants found performing swipes significantly easier than the other methods (Fig. 2.9). They also felt that performing swipes required less cognitive and physical demand than tilt and force, and caused less frustration (Fig. 2.9b). They found both stepper and swipe to be fast, accurate, and easy to use. There was no significant effect of method $\times$ setting on task completion time, which indicates towards the possibility that the performance of each method was similar across settings. This is interesting since prior studies reported performance decay when walking and interacting with a mobile device at the same time [38, 12]. This could be because of the slower pace of walking, and the use of a treadmill for walking since it did not require navigation. In real-world scenarios, users are forced to split their attention between the surroundings and the tasks on mobile devices to keep informed about the changing ambient environment [12]. Relevantly, a recent study with a similar experimental setup also failed to find a statistically significant difference in performance between input tasks in stationary and mobile settings [88]. On average, stepper was faster than tilt and force, however this difference was not statistically significant.

There was a significant effect of setting on actions per task, but no significant effect of method $\times$ setting. Which suggests that all methods suffered in terms of actions per task in mobile setting. Participants most likely performed more incorrect actions while walking, requiring corrective action, resulting in added actions per task. Interestingly, stepper took fewer actions than the other methods. Average actions per task for stepper, swipe, tilt, and force were 1.69 (SD = 1.2), 3.69 (SD = 3.5), 2.35 (SD = 1.5), and 4.81 (SD = 4.9), respectively. We speculate this is because, users spun the number wheel then waited until it slowed down, rather than a burst of repetitive spins. This also explains the higher task completion time for the method (Fig. 2.8).

The effects of method on the dependent variables yielded medium–large effect sizes, while the effect sizes of all statistically significant questionnaire data were large, indicating strong relationships between the examined variables. However, the effects of setting yielded a small effect size, hence we recommend caution in interpreting this result.

2.6 Experiment 2: Individual Vs. Inline

We conducted a second study to investigate whether the performance of the four methods differ when working with individual values and values embedded in text (i.e., inline values). The purpose was to find out whether the increased visual scan time and the physical and cognitive loads involved in editing inline numeric values affect the performance of the examined methods. Inline values usually require extra time to locate and select due to the surrounding text and the “fat-finger problem” [132], respectively. The study protocol was reviewed and approved by the Institutional Review Board (IRB). The study was completed before the World Health Organization declared the outbreak of COVID-19 a pandemic.

2.6.1 Apparatus

We used the same apparatus as the first study (Section 2.4.1).

2.6.2 Participants

Twelve participants voluntarily took part in the study. None of them participated in the pilots or the first study. Their age ranged from 20 to 32 years (M = 23.36, SD = 4.0). Two of them identified themselves as female and ten identified as male. Ten of them were right-handed, one was left-handed, and one was ambidextrous. All right-handed and ambidextrous participants chose to wear the device on their left hand and interact with the device using the right hand, while the left-handed participant chose to wear it on the right hand and use the other hand for interaction. They all were experienced smartphone users (M = 7.62 years of experience, SD = 1.9). Three of them were smartwatch owners (M = 1.8 years of ownership, SD = 0.8). Five of them also had experience using force as an input modality on Apple iPhone devices (M = 1.9 years of experience, SD = 1.7). They all received U.S. $10 for participating in the study.

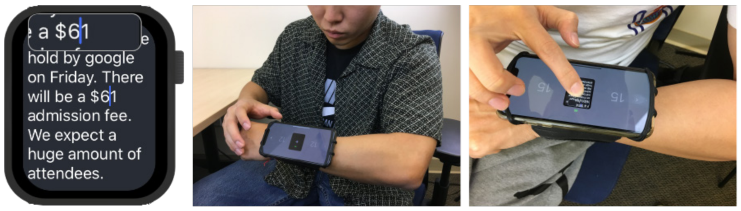

a $61
hold by google on Friday. There will be a $61 admission fee. We expect a huge amount of attendees.

Figure 2.10: A numeric value embedded in text (left) and two participants taking part in the study by changing an individual value and a value embedded in text, respectively (right).

2.6.3 Design

We used a within-subjects design for the study, where the independent variables were *placement* and *method*, and the dependent variables were the same performance metrics as the first study (Section 2.4.3). We also used the same questionnaires. In summary, the design was as follows.

12 participants $\times$

2 placements (individual and inline, *counterbalanced*) $\times$

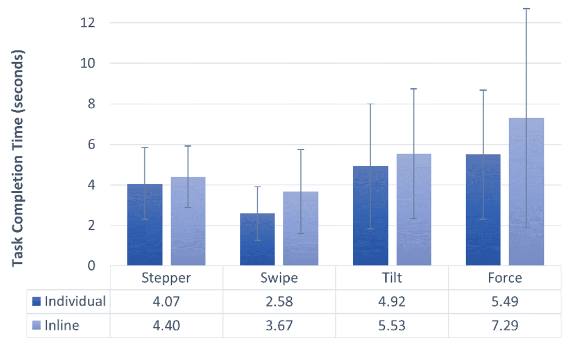
4 methods (the default stepper, swipe, tilt, and force, *counterbalanced*) $\times$

15 random two-digit values between 9 and 100

= 1,440 numeric values in total, excluding practices.

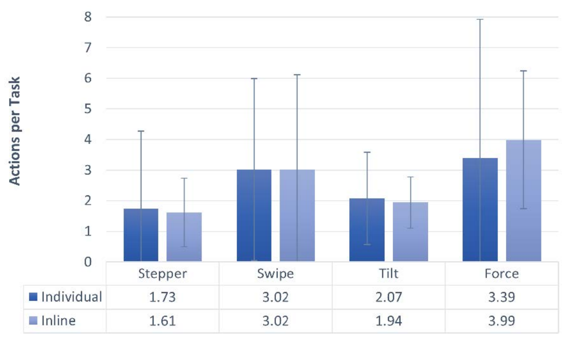
2.6.4 Procedure

The study used the same procedure (Section 2.4.6) and experimental tasks (Section 2.4.4) as the first study, with the *setting* independent variable replaced with *placement*.



	Stepper	Swipe	Tilt	Force
■ Individual	4.07	2.58	4.92	5.49
■ Inline	4.40	3.67	5.53	7.29

(a) Task completion time (seconds)



	Stepper	Swipe	Tilt	Force
■ Individual	1.73	3.02	2.07	3.39
■ Inline	1.61	3.02	1.94	3.99

(b) Actions per task

Figure 2.11: Average task completion time and actions per task for all methods in the two settings (individual and inline). Error bars represent ±1 standard deviation (SD).

2.6.5 Results

A Shapiro-Wilk test revealed that the response variable residuals are normally distributed. A Mauchly's test indicated that the variances of populations are equal. Thus, we used a repeated-measures ANOVA to analyze the quantitative data. In contrast, we used a Friedman test to analyze the questionnaire data. We also report effect sizes of all statistically significant results: eta-squared (η^2) for ANOVA and Kendall's W for Friedman test [11]. Eta-squared uses Cohen's interpretation [45] where 0.01 constitutes a small, 0.06 constitutes a medium, and > 0.14 constitutes a large effect. Kendall's W uses a different interpretation by Cohen [45], where 0.1 constitutes a small, 0.3 constitutes a medium, and > 0.5 constitutes a large effect. There were no significant effects of the order of conditions on the dependent variables ($p > .75$), which suggests that counterbalancing worked [101, pp. 177–180].

2.6.5.1 Task Completion Time

An ANOVA identified a significant effect of method on task completion time ($F_{3,33} = 69.88, p < .000005, \eta^2 = .23$)[6]. An ANOVA also identified a significant effect of placement ($F_{1,11} = 22.37, p = .0006, \eta^2 = .03$). There was also a significant method $\times$ placement interaction effect ($F_{3,33} = 4.60, p = .0081, \eta^2 = .01$). A Tukey-Kramer Multiple-Comparison test revealed that all methods were significantly different from one another. Swipe was significantly faster and force was significantly slower than all other methods. Figure 2.11a illustrates average task completion time for the four methods in both placements.

2.6.5.2 Actions per Task

An ANOVA identified a significant effect of method on actions per task ($F_{3,33} = 5.17, p = .004, \eta^2 = .12$). An ANOVA failed to identify a significant effect of placement ($F_{1,11} = 0.34, p = .57$). There was no method $\times$ placement interaction effect ($F_{3,33} = 1.55, p = .22$). A Tukey-Kramer Multiple-Comparison test revealed that stepper and force were significantly different from one another. Stepper required significantly fewer actions per task compared to force. The other two methods were comparable. Figure 2.11b illustrates average actions per task for the four methods in both placements.

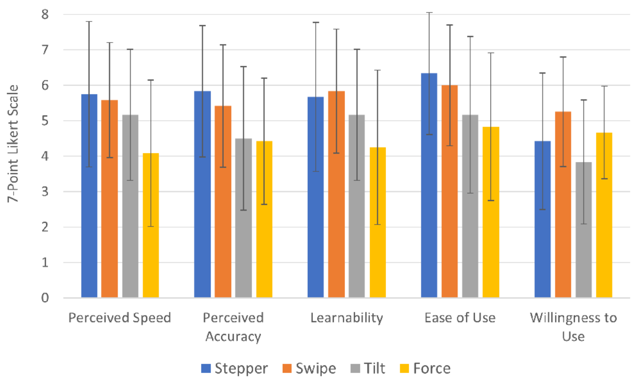
2.6.6 User Feedback

We used a non-parametric Friedman test to analyze the questionnaire data. Like the previous study, we present raw TLX scores by analyzing the sub-scales individually [71].

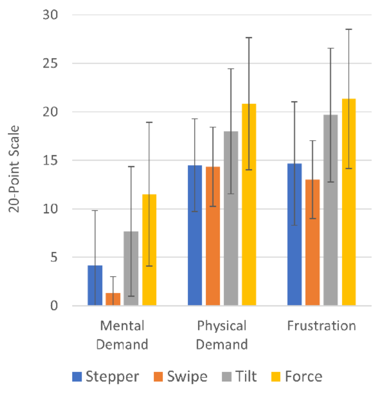
2.6.6.1 Perceived Performance and Preference

A Friedman test identified a significant effect of method on ease of use ($\chi^2 = 11.78, df = 3, p = .008, W = .32$). However, no significant effect was identified on perceived speed ($\chi^2 = 7.8, df = 3, p = .05$), perceived accuracy ($\chi^2 = 7.73, df = 3, p = .05$), learnability ($\chi^2 = 7.19, df = 3, p = .06$), or willingness to use ($\chi^2 = 10.07, df = 3, p < .05$). Figure 2.12a illustrates average user ratings of the four methods.

[6]The p value was too small for the NCSS and SPSS statistical software to display the exact value as they and most other statistical software display values up to six decimal places.



(a) Usability questionnaire



(b) Partial NASA-TLX questionnaire

Figure 2.12: Average user ratings of the four methods on a 7-point Likert scale, where “1” to “7” represented “strongly disagree” to “strongly agree” (left) and on NASA-TLX’s 20-point scale, where “1” to “20” represented “no demand” to “extreme demand” (right). Error bars represent ± 1 standard deviation (SD).

2.6.6.2 Mental and Physical Demand

A Friedman test identified a significant effect of method on mental demand ($\chi^2 = 11.64, df = 3, p = .009, W = .32$), physical demand ($\chi^2 = 16.10, df = 3, p = .001, W = .45$), and frustration ($\chi^2 = 12.57, df = 3, p = .006, W = .35$). Participants found picker and swipe to be the least mentally and physically demanding compared to tilt and force. They also found the former methods to be the least frustrating. Force was rated substantially higher in terms of mental and physical demand, as well frustration compared to the other methods. Figure 2.12b illustrates average user ratings of the four methods.

2.7 Discussion

The findings of this study are comparable to the first study. Swipe was significantly faster than the other methods with both individual and inline numbers. Average task completion time for stepper, swipe, tilt, and force were 4.23 (SD = 1.5), 3.13 (SD = 1.7), 5.23 (SD = 2.9), and 6.39 (SD = 4.3) seconds, respectively. Also, stepper and force required the least and the most number of actions, respectively. Average actions per task for stepper, swipe, tilt, and force were 1.67 (SD = 1.2), 3.02 (SD = 2.5), 2.01 (SD = 1.2), and 3.69 (SD = 4.5), respectively.

Qualitative results are also similar. Participants found both stepper and swipe significantly easier to use than the other methods. They found picker and swipe to be the least mentally and physically demanding compared to tilt and force. They also found swipe to be the least frustrating. These findings establishes the swipe-based method as a more effective input stepper for smartwatches. However, comparing Fig. 2.9b and Fig. 2.12b one can see that the methods were rated relatively poorly on NASA-TLX scale in the second study. This is expected since the inline placement of the numeric values required users to first locate and navigate to the value, then edit it, which required additional effort.

The effects of method on the dependent variables yielded medium–large effect sizes, while the effect sizes of all statistically significant questionnaire data were large, indicating strong relationships between the examined variables. However, the effects of placement yielded a small effect size, hence we recommend caution in interpreting this result.

2.7.1 Design Recommendations

Based on the results of the two studies and user feedback, we recommend using a hybrid of input stepper and the swipe-based method to enable selection of numbers with short edit distances with flicks (e.g., changing "12:15 pm" to "12:30 pm") and long edit distances with swipes and swipe-and-hold gestures (e.g., changing "15%" to "70%"). We also recommend automatically slowing down the spinning rate when the number reaches a probable value for easier selection. A prior work [111] showed that it is often possible to predict the intended value through contextual awareness and even by using simple rules and patterns (for example, "12:30 pm" is more probable than "12:22 pm"). It may also be effective to enable users to select the method they prefer the most for picking numbers.

2.8 Generalizability

Although it is relatively common to use smartphones or tablets to study interactions with smartwatches because of the technological limitations of current smartwatches (e.g., [41, 110, 94]), due to the absence of empirical evidence, it is unclear whether the performance recorded on a simulated smartwatch is generalizable to actual smartwatches. Hence, to increase the external validity of the work, we replicated not only the interface but also the holding position and posture of a smartwatch (Fig. 2.6). In all evaluations, participants wore the simulated smartwatch (i.e., the smartphone) on their wrist and interacted with it as one would with an actual smartwatch. Relevantly, a prior work reported that text entry performances with similar sized virtual keyboards on an actual smartwatch and a simulated smartwatch on a smartphone were not significantly different in terms of entry speed and accuracy [140].

2.9 Conclusion

We presented three new methods for number picking on smartwatches by performing directional swipes, twisting the wrist, and varying contact force on the screen. Unlike the default number picker, the proposed methods enable users to actively switch between slow-and-steady and fast-and-continuous increments and decrements. We evaluated these methods in two user studies, exploring stationary vs. mobile settings and individual vs. inline number editing, respectively. In both studies, the swipe-based method yielded a significantly faster input rate. Participants also found the method fast, accurate, and the least mentally and physically demanding. Accuracy rates were comparable among the conditions. These results establish swipe as an effective number picking method on smartwatches.

In the furture, we will evaluate the proposed number pickers on larger touchscreen-based devices, particularly smartphones and tablets. We will also design additional methods that exploit the crown and the bezel of a smartwatch for number picking.

Chapter 3

Force-Based Input on Smartwatches

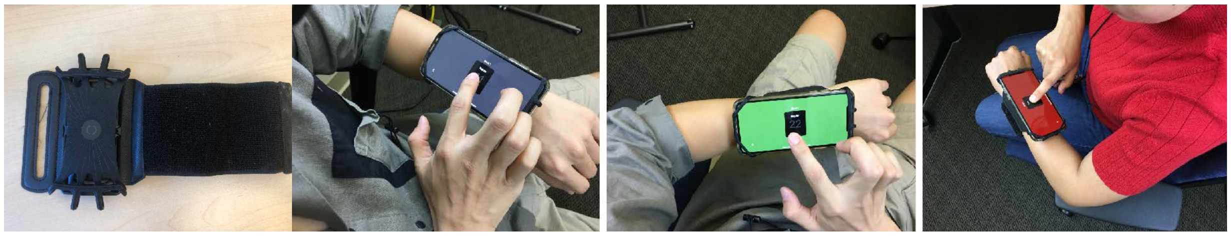
3.1 Introduction

Smartwatches are becoming increasingly popular among mobile users [90]. However, selecting targets, particularly small targets, is difficult on smartwatches due to the "fat-finger problem" [132]. To facilitate precise target selection, most smartwatch applications either clutter the interface by using large interactive elements or require users to perform a sequence of actions. Both of these approaches affect performance and user preference [82, 116]. In this work, we propose a one-directional force-based target selection approach, with which users slide the finger closer to the target, then variate contact force to move the cursor along the x-axis (reducing the force moves the cursor to the left and increasing the force moves the cursor to the right) to select the target.

Toward this, we first identify the most comfortable range of force on smartwatches. We then compare one-directional force with traditional touch in a Fitts' law experiment. Finally, to demonstrate practical usage of the proposed method, we design and evaluate two new force-based text entry techniques for smartwatches (Section 3.6). Unlike the existing techniques, these neither occupy most of the display nor use aggressive correction model by disabling character-by-character input. The contribution of this work is thus threefold: identifying the most comfortable range of force on smartwatches, comparison of one-directional force-tap with conventional touch in a Fitts' law study, and the design and evaluation of two new novel force-based keyboards. All studies reported here were approved by the Institutional Review Board (IRB) and conducted abiding by the institute's COVID-19 preventive measures.

3.2 Related Work

Not much work has focused on target selection on smartwatches. Hara, Umezawa, and Osawa [69] investigated the effects of button size and location on target selection performance with the index finger. They found out 5 mm and 7 mm targets are susceptible to significantly higher selection errors than 10 mm targets. Ishii and Shizuki [79] developed eight different callout features that display and magnify the area occluded by the finger in a non-occluded area. In multiple evaluations, participants found these methods to be useful in target selection. Xia, Grossman, and Fitzmaurice [136] developed a finger-mounted fine-tip stylus to enable fast and accurate pointing with almost no occlusion. In a study, the stylus reduced erroneous selection by 80% compared to traditional touch



(a) The wristband (b) The custom Apple iOS application

Figure 3.1: (a) The wristband used in the study and (b) participants applying force on the screen. The grey, green, and red background colors indicate the initial force level, and correct and incorrect changes in the force level, respectively.

interaction. Yeo et al. [139] compared target selection in all directions with force-tap, twist, and pan gestures, where force-tap was the most challenging of all methods since it was difficult to apply force in the correct direction. Kurosawa, Sakamoto, and Ono [89] used a tilt and force hybrid method for target selection on a smartwatch. This method uses an electromyography sensor on the arm to detect tilting of the device. To select a target, users first tilt the hand to indicate the cursor direction, then apply force on the arm to move the cursor to the target. Darbar, Sen, and Samanta [48] augmented a smartwatch with four pressure sensors to enable users to apply different levels of force on the two sides of the device for zooming, scrolling, and rotating an interactive map. Ahn et al. [1] used a pressure-sensitive wristband to perform similar interactions. These three methods, however, require extramural hardware to function.

There has been some research on force-based text entry on smartphones. McCallum et al. [102] developed a force-based technique for the standard 12-key mobile keypad by utilizing three levels of force. Likewise, Tang, Beebe, and Kramer [125] developed a three-key chorded keyboard with three force levels. Both these methods were highly error prone, yielding 9% and 18% error rates in user studies, respectively. Brewster and Hughes [30] presented several pressure-based techniques to switch between uppercase and lowercase letters on a virtual Qwerty, some of which were faster and more accurate than the Shift key. Arif and Stuerzlinger [18] and Arif, López, and Stuerzlinger [13] developed an error prevention technique that requires applying extra force on the keys to enter the less probable letters. The effects of the approach on text entry performance were contradictory in two consecutive studies. Arif and Stuerzlinger [19] enabled bypassing auto-correction by applying extra force on the keys. In an evaluation, this approach significantly improved text entry speed and accuracy. Vertanen et al. [131] used a similar approach on smartwatches. Zhong et al. [144] developed a one-dimensional alphabetical keyboard with a sliding cursor over the letters. The cursor covered multiple letters. Users moved the cursor by variating contact force, confirmed selection by performing a quick release (reducing pressure quickly without lifting the finger), the keyboard then disambiguated the input using a probabilistic model. It also enabled entering one character at a time by using a multi-tap [80] like approach. The keyboard displayed the selected letters in descending order of probability, users then multi-tapped on the screen to select the intended letter. In an evaluation, the word- and character-level approaches yielded on average 4 and 11 wpm, respectively, on a smartphone. Chapater 2 (Chap. 2) also presented a force-based approach to pick numbers from a number wheel on smartwatches. There are, however, no force-based character-level text entry methods available for smartwatches.

3.3 User Study 1: Levels of Force

We conducted a user study to investigate the levels of force users can comfortably apply on smartwatches. The purpose was to map the most comfortable range of force to cursor movements on a tiny display.

3.3.1 Participants

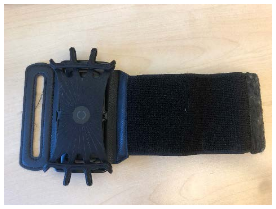
Thirteen participants took part in the study. Their age ranged form 24 to 34 years (M = 28.5, SD = 3.4). Three of them identified themselves as women and ten as men. Nine of them were right-handed and four were left-handed. All of them were experienced mobile users (M = 8.4 years, SD = 2.3). Five of them also owned a smartwatch (M = 0.5 years, SD = 0.9). Six of them had experience with force-based interaction through Apple iOS's 3D touch [29]. They all received U.S. $10 for participating in the study.

3.3.2 Apparatus

We used an iPhone X (43.6$\times$70.9$\times$7.7 mm, 174 grams) running on iOS version 12.1 at 1125$\times$2436 pixels resolution in the user study. We developed a custom app using the default iOS SDK to simulate an Apple Watch 5's 740 mm^2 display area (312$\times$390 pixels) on the smartphone. We made the surrounding area of the simulated smartwatch touch-insensitive to avoid the effects of accidental touches during the study. We used a smartphone instead of an actual smartwatch since current smartwatches do not provide the support for continuous force detection. Apple Watch detects only the absence and presence of extra force. It is relatively common to use larger devices to study interactions with smartwatches due to technological limitations of current smartwatches [41, 110, 94, 79]. Relevantly, a prior work reported that text entry performances of a keyboard on an actual smartwatch and a simulated smartwatch on a smartphone were comparable in terms of speed and accuracy [140]. To increase the external validity of the work, we replicated not only the interface but also the holding position and posture of a smartwatch. We used a wristband with silicone phone holder (55.5 grams) to attach the smartphone to the wrist of the participants like a smartwatch (Fig. 3.2a). The wristband held the device on the wrist firmly, thus participants did not have to hold it steady with the fingers of the other hand, although we noticed a few participants occasionally doing that. The holder was 180° rotatable but we did not enable participants to rotate the device during the study to eliminate a potential confound.

3.3.3 Design & Procedure

First, the participants signed the informed consent form and completed a demographics and mobile usage questionnaire. We then explained the research and demonstrated the custom app. Participants were instructed to wear the device on their non-dominant hand and interact using the other hand. All participants were seated, but were instructed not to rest their arms on the desk to increase the external validity of the study. The app displayed one force level on the screen. Participants were instructed


(a) The wristband

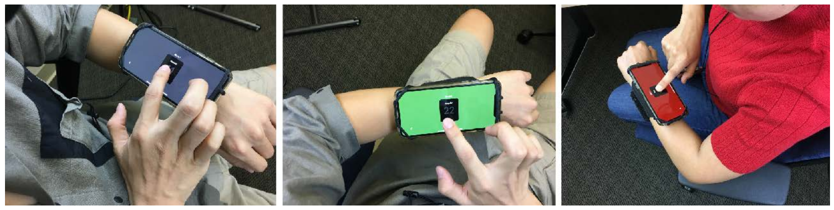
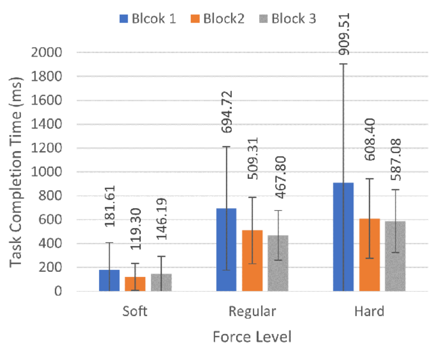

(b) The custom Apple iOS application

Figure 3.2: (a) The wristband used in the study and (b) participants applying force on the screen. The grey, green, and red background colors indicate the initial force level, and correct and incorrect changes in the force level, respectively.

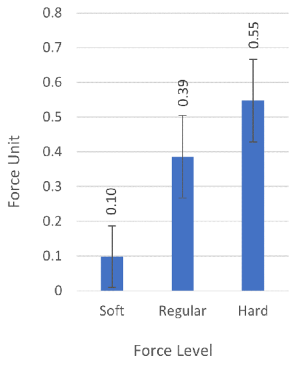
to touch the screen, then chance contact force as as instructed on the display, without compromising physical comfort. The area surrounding the simulated smartwatch screen was inactive, but was used to provide feedback on the applied level of force by changing the background color. Initially, the background color was grey, but changed to green when participants changed contact force correctly (reduced force for soft) and to red when changed contact force incorrectly (reduced force for hard) (Fig. 3.2b). Correct and incorrect force levels were determined based on the increments and decrements of the force value rather than predetermined thresholds for different levels. The system identified a correct input when the force value changed in accordance to the displayed force level. For example, for hard force level, the system registered a correct input when the force value was gradually increasing. The system ignored slight variations in the force level (abrupt, discontinuous changes) since it is almost impossible for users to maintain a constant level of force. Once done with one level, the app displayed the next force level. This process continued until all tasks of a block were completed. There were three block, each containing (3 levels $\times$ 18 tasks) 54 tasks. Participants were asked to take $\sim$5 minutes break before starting the next block to mitigate any discomfort due to variating contact force. In summary, the design was: 13 participants $\times$ 3 blocks $\times$ 3 levels (soft, regular, hard) randomized $\times$ 18 tasks $=$ 1,755 data points in total. Upon completion, participants took part in a brief interview discussing their experience in the study.

3.3.4 Results & Discussion

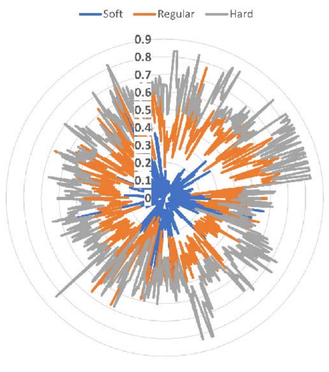
The default Apple iOS SDK returns a value between 0 and 6.67 for the amount of force imparted by the user's finger onto the screen. Similar to Chapter 2, we normalized it to the interval from 0 to 1 by dividing the received force value by the maximum force (6.67) for better presentation. The median force applied for soft, regular, and hard tasks were 0.10 (SD = 0.08), 0.39 (SD = 0.12), and 0.55 (0.12), respectively (Fig. 3.3b). We, thus, decided to use [0.05, 0.80] as the rage of our mapping function, where the lowest value is $\sim \frac{1}{2} \times SD$ from the median of soft force level and the highest value is $\sim 2 \times SD$ from the median of hard force level. These values were picked by closely studying the force patterns of all participants. While participants were fairly consistent in the minimum levels of force applied on the screen, their maximum levels of force varied. In about 10% of all incidents, participants applied a maximum force level closer to 0.8. This suggests that they are comfortable with this force level. Participants also confirmed this in the post-study interview. This encouraged us to increase the maximum value to offer more granularity in the proposed force-



(a) Task completion time (ms)



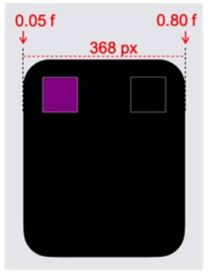
(b) Median force per level



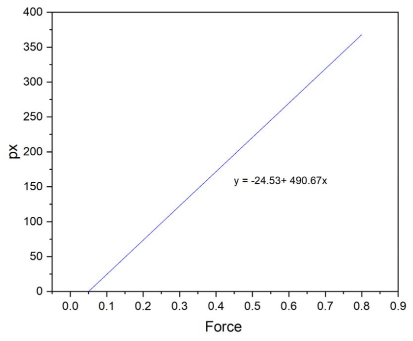
(c) Force applied in all tasks

Figure 3.3: (a) Average task completion time (ms) per block for each force level, (b) median force applied per level, and (c) a radar chart showing force applied in each task. Error bars represent ± 1 standard deviation (SD).

based selection method. In Fig. 3.3c, one can see that force values rarely went outside this range. The values between the range were then mapped to the 368 px horizontal space of the smartwatch using a linear function (Fig. 3.4). There was a significant effect of level on task completion time ($F_{2,12} = 63.94, p < .0001$). On average, soft, regular, and hard tasks took 149.03 ms (SD = 171.2), 557.28 ms (SD = 373.2), and 701.67 ms (SD =643.5) to complete, respectively (Fig. 3.3a).



(a) Smartwatch screen



(b) The linear mapping function

Figure 3.4: The force values within the range [0.05, 0.80] are mapped to the pixels of the display using a linear function.

3.4 1D Fitts' Law Protocol

Fitts' law is a well-established method for evaluating target selection on computing systems [100]. In the 1990s, it was included in the ISO 9241-9 standard for evaluating non-keyboard input devices by using Fitts' throughput as a dependent variable [123]. Most one-dimensional (1D) Fitts' law experiments combine serial responses with 1D movements. The targets of width W are placed on the two sides of the display (Fig. 3.5b). The target to select is highlighted. Once selected, the highlight moves to the opposite target. This back-and-forth selection continues until all targets are selected. Each movement covers an amplitude A, which is distance to the centre of the target (Fig. 3.5a). A *trial* is defined as one target selection task, whereas completing all tasks with a given amplitude is defined as a *sequence*. Throughput cannot be calculated on a single trial because a sequence of trials is the smallest unit of action in ISO 9241-9. Traditionally, the difficulty of each trial is measured in bits using an index of difficulty (ID), calculated as follows:

$$ID = log_2(\frac{A}{W} + 1) \tag{3.1}$$

The movement time (MT) is measured in seconds for each trial, then averaged over the sequence of trials. It is then used to calculate the performance throughput (TP) in bits/second (bps) using the following equation:

$$TP = \frac{ID}{MT} \tag{3.2}$$

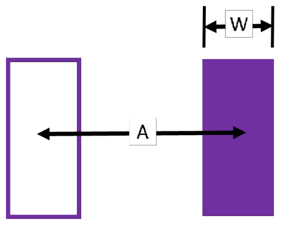
The revised ISO 9241-9 (9241-411) [78] measures throughput using an effective index of difficult ID_e, which is calculated from the effective amplitude A_e and the effective width W_e to make sure that the real distance traveled form one target to the next is measured. It also takes into account the spread of selections about the target center.

$$TP = \frac{ID_e}{MT} \tag{3.3}$$

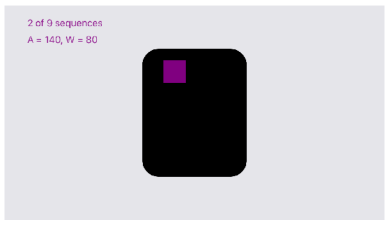
$$ID_e = log_2(\frac{A_e}{W_e} + 1) \tag{3.4}$$

The effective amplitude is the real distance travelled by the participants, while the effective width is calculated as follows, where SD_x is the standard deviation of the selection coordinates projected on the x-axis for all trials in a sequence. This accounts for any targeting errors by the participants, assuming that participants were aiming at the center of the targets.

$$W_e = 4.133 \times SD_x \tag{3.5}$$



(a) The 1D Fitts' law task in ISO 9241-9



(b) The custom Apple iOS application

Figure 3.5: (a) The target is highlighted in purple, (b) the custom Apple iOS app displaying $A = 140$, $W = 80$. It uses the same selection sequence as ISO 9241-9.

3.5 User Study 2: 1D Fitts' Law Study

We conducted a Fitts' law study to compare target selection performance of tap and force-tap. The purpose was to investigate if force-tap could be effective in selecting small targets on smartwatches. We focused only on movements in the x-axis because we envision force as a companion of touch rather than an independent selection method, where users tap in the proximity of a target then move the cursor to the left or right for precise selection. Besides, the difficulties in applying force in all directions is evident in a prior work [139], where force was significantly slower (2,600 ms) and more error prone (1.6%) than twist and pan gestures.

3.5.1 Participants

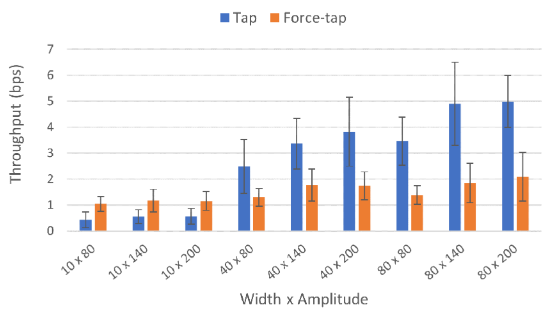
Twelve participants took part in the study. Their age ranged from 20 to 34 years (M = 29.3, SD = 2.3). Five of them identified themselves as women and seven as men. Ten of them were right-handed and two were left-handed. They all were experienced mobile device users (M = 10.8 years, SD = 2.7). Nine of them had experience with force-based interaction through Apple iOS's 3D touch [29]. They all received U.S. $15 for participating in the study.

3.5.2 Apparatus & Design

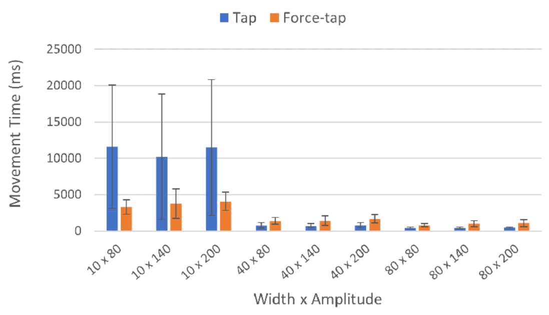
The study used the same apparatus as the previous study (Section 3.3.2). A custom app was developed carry out the 1D Fitts' law protocol described above. The experiment was a $2 \times 3 \times 3$ within-subjects design. The independent variables were method (tap, force-tap), amplitude: 80, 140, 200 px (20, 35, 50 mm), and width: 10, 40, 80 px (1.5, 12.5, 20 mm). There were 20 trials per condition. The amplitudes were selected based on the display area, to make sure that the targets do not overlap or go outside the boundary. The widths were selected based on the optimal widths recommended in prior research and design guidelines [69, 70, 77, 74]. The dependent variables were throughput (*TP*) and movement time (*MT*).

3.5.3 Procedure

The study used the same procedure as the previous study (Section 3.3.3), except for the tasks, which were in accordance with the 1D Fitts' law protocol discussed in Section 3.4. In the study, participants selected targets using the two selection methods in a counterbalanced order. The cursor was initially positioned in between the targets, then moved back-and-forth from one target to another by either tap or variating contact force. Participants were instructed to select the targets as fast as possible. Incorrect selections were not allowed, in such cases, participants had to select it again. We enforced a $\sim$5 minute break after a condition to avoid the effect of fatigue. After the completion of both conditions, participants completed the NASA-TLX questionnaire [20] to rate the perceived workload of the methods. They also took part in a brief interview session to discuss their experience in the study.



(a) Throughput (bps)



(b) Movement time (ms)

Figure 3.6: Average throughput (bps) and movement time (ms) for both methods across all examined widths and amplitudes. Error bars represent ± 1 standard deviation (SD).

3.5.4 Results

3.5.4.1 Throughput

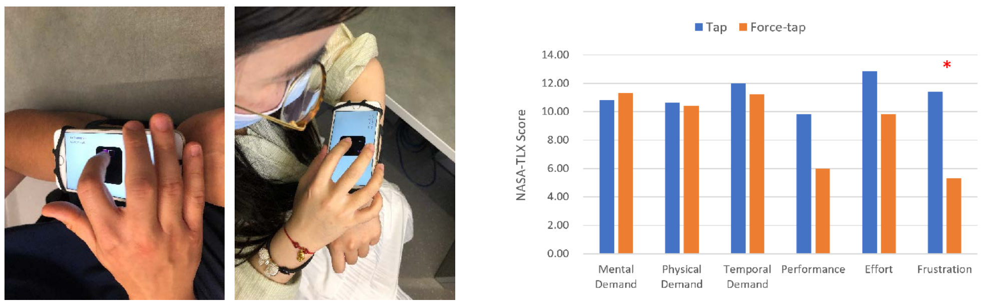
An ANOVA identified a significant effect of method on throughput ($F_{1,11} = 76.05, p < .0001$). The average throughput for tap and force-tap were 2.7 bps (SD = 1.8) and 1.5 bps (SD = 0.4), respectively. An ANOVA also identified significant effects of width ($F_{2,11} = 165.50, p < .0001$) and amplitude ($F_{2,11} = 24.56, p < .0001$). The *method* $\times$ *width* interaction effect was also statistically significant ($F_{2,22} = 78.12, p < .0001$). However, the *method* $\times$ *amplitude* interaction effect was not statistically significant ($F_{2,22} = 3.15, p = .06$). Fig. 3.6a illustrates average throughput for both methods across all examined widths and amplitudes.

3.5.4.2 Movement Time

An ANOVA identified a significant effect of method on movement time ($F_{1,11} = 5.92, p < .05$). The average movement time for tap and force-tap were 4,098 ms (SD = 5,266) and 2,052 ms (SD = 1,282), respectively. There was also a significant effect of width ($F_{2,11} = 30.78, p < .0001$). However, no significant effect of amplitude was identified ($F_{2,11} = 0.69, p = 0.51$). The *method* $\times$ *width* interaction effect was also statistically significant ($F_{2,22} = 11.02, p < .0005$). But the *method* $\times$ *amplitude* interaction effect was not statistically significant ($F_{2,22} = 0.96, p = 0.4$). Fig. 3.6b illustrates average movement time for both methods across all examined widths and amplitudes.

3.5.4.3 Perceived Workload

We present raw TLX scores by analyzing the sub-scales individually, which is a common modification made to NASA-TLX [72]. A Wilcoxon Signed-Rank test failed to identify significant effects of method on mental demand ($z = -0.18, p = .86$), physical demand ($z = -0.1, p = .92$), temporal demand ($z = -0.23, p = .81$), performance ($z = -1.65, p = .10$), and effort ($z = -1.25, p = .21$). However, a significant was identified on frustration ($z = -2.52, p < .05$). Fig. 3.7c illustrates average NASA-TLX ratings of the two methods.



(a) Tap (b) Force-tap (c) Average NASA-TLX scores

Figure 3.7: Two participants selecting targets with (a) tap and (b) force-tap, and (c) average perceived workload of the two methods. The red asterisk indicate statistically significant difference.

3.5.5 Discussion

Tap yielded significantly higher throughput than force-tap ($\sim$80% higher). However, the interaction effects suggest that throughput of the two methods were significantly affected by the target size and amplitude. A post-hoc Tukey-Kramer test revealed that force-tap performed significantly better with the smallest target ($\sim$120% higher throughput), while tap performed significantly better with the bigger ones ($\sim$100–151% higher throughput). This is also evident in Fig. 3.8 that illustrates average throughput for the two selection methods with the three target sizes (10, 40, 80 pixels) fitted to power trendlines. As one can see, both selection methods conformed to Fitts' law, as expected — with both, throughput decreased linearly with decreasing target sizes (tap: $R^2 = 0.91$, force-tap: $R^2 = 0.88$), however, tap had a much steeper drop than force-tap.

Overall, force-tap was significantly faster than tap ($\sim$50% faster). A deeper analysis indicated that difficulties in selecting the smallest target contributed to this. A post-hoc Tukey-Kramer test revealed that selection time for the smallest target was significantly slower with tap than with force-tap

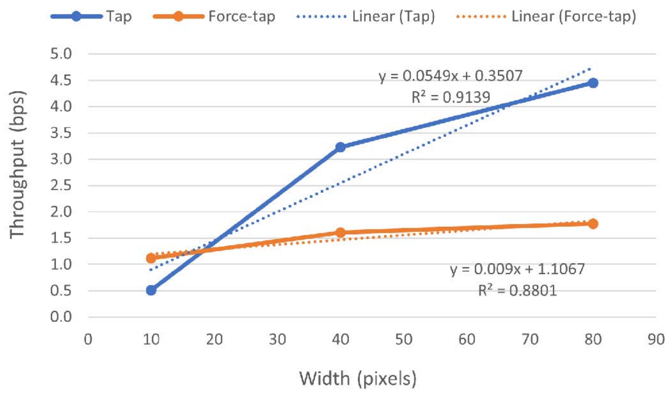
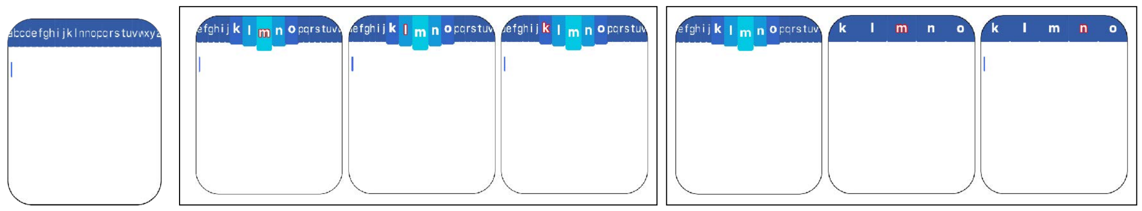


Figure 3.8: Average throughput (bts) for the two selection methods with the three target sizes fitted to power trendlines.

(∼67% slower), while selection time for the larger targets were somewhat comparable (see Fig. 3.6b). These suggest that force could be an effective method for selecting smaller targets on tiny displays. Post-study questionnaire and interview also support this. Participants found both methods relatively similar in terms of mental, physical, and temporal demands, performance, and effort, but were significantly more frustrated with tap than force-tap (see Fig. 3.7c), primarily due to the smaller targets. One participant (male, 27 years) commented, *"Some blocks have very tiny boxes and it was very hard to hit the right place. It was frustrating."* Another participant (female, 29 years) said, *"Both touch and force seemed the same for bigger squares."* Participants also commented on the learnability of force-tap. One participant (male, 34 years) stated, *"Force is pretty novel to me, it took me a little time to get used to the smallest one. Once I got used to it, I could finish the task faster than beginning."* Some participants, on the other hand, preferred using force-tap exclusively on smartwatches. One participant (female, 27 years) commented, *"It is much easier for me to complete the task of force than touch."* Based on the findings, we recommend enabling both tap and force-tap on smartwatches. Tap is fast and reliable for bigger targets, but for those occasional smaller targets, force-tap is much more reliable, faster, and causes less frustration.



(a) Initial state (b) Slide-force keyboard (c) Slide-force-slide keyboard

Figure 3.9: (a) The initial state of the keyboards showing all letters from 'a' to 'z', (b) the process of entering the letter 'k' with the slide-force keyboard: the user slides her finger anywhere on the screen in relevance to the target, the keyboard magnifies the five nearby keys, because the target is on the left, the user reduces contact force to highlight the letter, then releases touch to enter it, (c) the process of entering the letter 'n' with the slide-force-slide keyboard: the user slides her finger anywhere on the screen in relevance to the target, she applies extra force to replace the keyboard with magnified versions of the five nearby keys, the user then slides her finger over the target letter and releases touch to enter it.

3.6 Slice Keyboards

To demonstrate practical usage of force input, we designed two <u>sli</u>de and for<u>ce</u> (slice) keyboards that enable users to enter one character at a time using force-tap and finger slide gestures. We used text entry as our test scenario because entering text is an extreme case of target selection, requiring repetitive selection of different targets (the keys). We designed two alphabetical slice keyboards that display all letters of the English alphabet in a 368×70 px row (Fig. 3.9a). To enter a letter, the user slides her finger horizontally along the x-axis anywhere on the screen. The letter closest to the x-coordinate of the finger and the two neighbouring letters from each side (five in total) are magnified using a zoom-in effect. The <u>slide-force</u> keyboard highlights the central letter (Fig. 3.9b).

The user can reduce contact force to highlight the left letters or increase contact force to highlight the right letters. Realising touch enters the highlighted letter. The slide-force-slide keyboard does not highlight the magnified letters (Fig. 3.9c), instead requires the user to apply extra force to replace the keyboard with magnified versions of the five letters (73.6 $\times$ 70 px each). The initial zoom-in mode of both keyboards are identical. But since the slide-force-slide keyboard enables target selection via both force and slide, the candidate five letters are displayed on the keyboard area to facilitate sliding. Initially, the central letter is highlighted, but the user can side her finger over any key, then release touch to enter the corresponding letter. Both keyboards enable the entry of space and backspace by performing swift left and right strokes, respectively, anywhere on the screen.

3.7 User Study 3: Slide-Force v. Slide-Force-Slide

We conducted a user study to compare the two keyboards. Apart from evaluating the new keyboards, one purpose of the study was to demonstrate that force-based selection method could be used in practical scenarios.

3.7.1 Participants

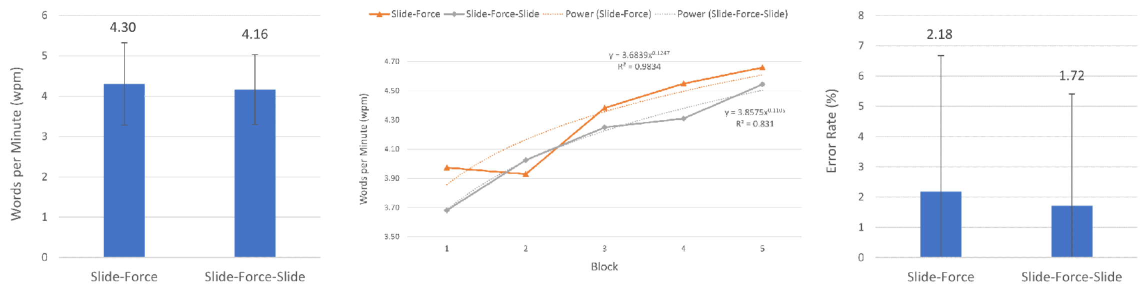
Ten participants took part in the study. Their age ranged from 24 to 34 years (M = 28.2, SD = 2.7). None of them participated in the previous studies. Three of them identified themselves as women and seven as men. Eight of them were right-handed and two were left-handed. They all were experienced mobile device users (M = 10 years, SD = 1.3). Six of them also owned a smartwatch (M = 1.6 years, SD = 1.9). None of them had prior experience with force-based interaction. They all received U.S. $15 for participating in the study.

3.7.2 Apparatus & Design

The study used the same apparatus as the previous studies (Section 3.3.2). It was a 2 $\times$ 5 within-subjects design. The independent variables were method (slide-force, slide-force-slide) and block. There were five short English phrases [130] per block. The dependent variables were the commonly used words per minute (wpm) and error rate (%) performance metrics in text entry research [17]. In summary, the design was: 10 participants $\times$ 2 methods $\times$ 5 blocks $\times$ 5 phrases = 500 phrases in total.

3.7.3 Procedure

The study used the same procedure as the first study (Section 3.3.3), except for the tasks. During the study, participants transcribed short English phrases from a set [130] using the two keyboards in a counterbalanced order. A custom app displayed one random phrase at a time outside the smartwatch area (Fig. 3.11). Participants were instructed to read and memorize the phrase, then transcribe it as fast and as accurate as possible using either of the method. Error correction was recommended but not forced. Once entered, the app automatically displayed the next phrase. This continued until all phrases were entered. We enforced a $\sim$5 minute break between the conditions to avoid the effect of fatigue. We enabled participants to practice with the methods by entering five phrases with each technique before the corresponding condition. These phrases were not repeated in the main study. After the study, participants completed a custom questionnaire to rate the performance of the methods. They also took part in a brief interview session to discuss their experience in the study.



(a) Entry speed (wpm) (b) Entry speed (wpm) per block (c) Error rate (%)

Figure 3.10: (a) Average entry speed (wpm) of the two methods, (b) average entry speed (wpm) across all blocks fitted to power trendlines, (c) average error rate (%) of the two methods. Error bars represent ±1 standard deviation (SD).

3.7.4 Results

3.7.4.1 Entry Speed

An ANOVA failed to identify a significant effect of method on entry speed ($F_{1,9} = 0.28, p = .61$). On average slide-force and slide-force-slide yielded 4.3 wpm (SD = 1.0) and 4.2 wpm (SD = 0.7), respectively (Fig. 3.10a). However, there was a significant effect of block ($F_{4,9} = 30.12, p < .0001$). Fig. 3.10b illustrates average entry speed of the two methods across all blocks.

3.7.4.2 Error Rate

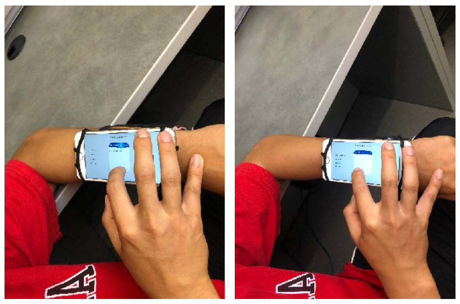
An ANOVA failed to identify a significant effect of method on error rate ($F_{1,9} = 0.86, p = .38$). On average SF and SFS yielded 2.2% (SD = 4.5) and 1.7% (SD = 3.7) error rates, respectively (Fig. 3.10c). There was also no significant effect of block ($F_{4,9} = 1.39, p = .26$).

3.7.4.3 User Feedback

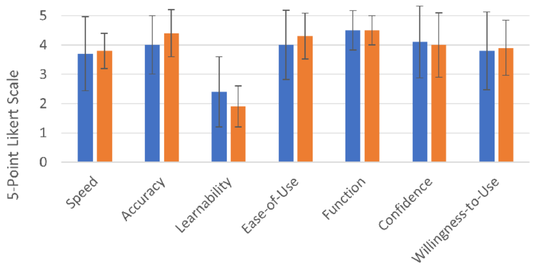
A Wilcoxon Signed-Rank test failed to identify significant effects of method on perceived speed ($z = -0.14, p = .89$), accuracy ($z = -1.26, p = .21$), learnability ($z = -1.29, p = .20$), ease-of-use ($z = -1.0, p = .32$), functionality of the features ($z = 0, p = 1.0$), confidence in using the methods ($z = -0.33, p = .74$), and willingness to use the methods on their smartwatches ($z = -0.21, p = .83$). Fig. 3.11c illustrates average user ratings of the two methods.

3.7.5 Discussion

Entry speed of the two methods were comparable. Besides, the 4.3 wpm is much lower than the performance of the existing character-level methods, which were reported to yield between a 4.3 and 19.6 wpm entry speed [15]. These methods, however, used much larger keys by occupying up to



(a) Slide-Force (b) Slide-Force-Slide



(c) Average user ratings

Figure 3.11: A participant entering text with (a) slide-force and (b) slide-force-slide keyboards, and (c) average user ratings of the two methods on a 5-point Likert scale (1–5: low–high). Error bars represent ±1 standard deviation (SD).

85% of the screen [114] and/or evaluated in longer sessions and on much larger smartwatches. It is also important to note that, we observed a significant effect of block on entry speed. Words per minute with both techniques improved substantially in the last block compared to the first (17% and 23% improvements with slide-force and slide-force-slide, respectively). A post-hoc Tukey-Kramer test identified these differences to be statistically significant. Fig. 3.10b illustrates average entry speed of the two methods in all blocks fitted to power trendlines, where one can see that both slide-force ($R^2 = 0.98$) and slide-force-slide ($R^2 = 0.83$) correlated well with the power law of practice [34]. These suggest that learning occurred with both methods even in the short duration of the study, thus possible that performance with the methods will improve further with practice. Many participants also felt that their performance improved with practice. One participant (male, 30 years) commented, *"It took some time to get to used to it, but after that it was easy to use."* There was no significant effect of method on error rate. Both methods were fairly accurate with about 2% error rate, which is much lower than the 5–28% error rate reported for the existing character-level methods for smartwatches [15]. Participants were mostly indifferent about the two methods in the post-study questionnaire (Fig. 3.11c). However, we noticed that participants were split about which method they preferred the most. One participant (female, 29 years) commented, *"The split-force-split was the fastest for me to use even through I hadn't used it before."*, while another (male, 34 years) said, *"To sum up, slide-force is the best way to type for me."* However, many participants found both methods difficult to learn. One participant (female, 24 years) commented, *"They [both] are quite hard to control how much force need to push when choose the letters."* These results suggest that while the keyboards may not be appropriate as the primary method of text entry on smartwatches, these could be used as extensions to the primary input method (which are predominantly predictive with aggressive correction model, thus do not always enable out-of-vocabulary word entry [114]) for non-dictionary word entry or to enter short phrases (e.g., short response to a text message). Most importantly, the fact that force input prevailed even in an extreme scenario like text entry indicate that it can be effectively used an active mode of interaction on smaller devices.

3.8 Conclusion

In this work, we investigated the possibility of using contact force as an active mode of interaction on smartwatches, especially to enable the selection of smaller targets. We presented the results of three user studies. The first identified the most comfortable range of force users can apply on smartwatches. The second revealed that force input is significantly more effective in selecting smaller targets than touch. Finally, the third study demonstrated that force could be effectively used in practical scenarios by developing and comparing two new force-based character-level text entry techniques. In addition to the means for demonstrating force input's effectiveness, we see these keyboards as independent contributions as they have much smaller footprints than the existing character-level methods and users were relatively fast at learning these. We envision these keyboards being used as extensions to predictive keyboards that disable out-of-vocabulary word entry due to their aggressive correction models, to enable the entry of occasional non-dictionary words.

One limitation of the work is the studies reported here were conducted on simulated smartwatch interfaces on a smartphone. While this is fairly common in the literature [41, 110, 94, 79] (also discussed in Chapter 2), due to the absence of empirical evidence, it is unclear if the performance recorded on a simulated smartwatch is generalizable to actual smartwatches. In the future, we will explore different control-display mapping functions for force input. We will also use machine learning approaches to make the method more reliable. We also hope to evaluate the new keyboards in longitudinal studies and augment them with predictive systems for faster text entry.

Chapter 4

Editing & Formartting on Smartphones

4.1 Introduction

Text entry has become a vital part of our everyday life. Nowadays, we enter text not only on desktop computers but also on the go on our mobile devices. While there is a rich body of work on text entry and error correction on mobile devices, not much work focused on text editing or formatting. However, with the increased use of mobile devices, developing efficient text editing and formatting approaches are the next integral step in the progression of mobile text entry. Besides, text editing requires effective approaches for precise cursor positioning and text selection, which can also benefit error correction. This work defines text *editing* as the process of manipulating existing text with modeless editing operations, including cut, copy, paste, and move [126], not to be confused with *error correction* that involves correcting incorrect text or *revision* that involves rewriting parts of

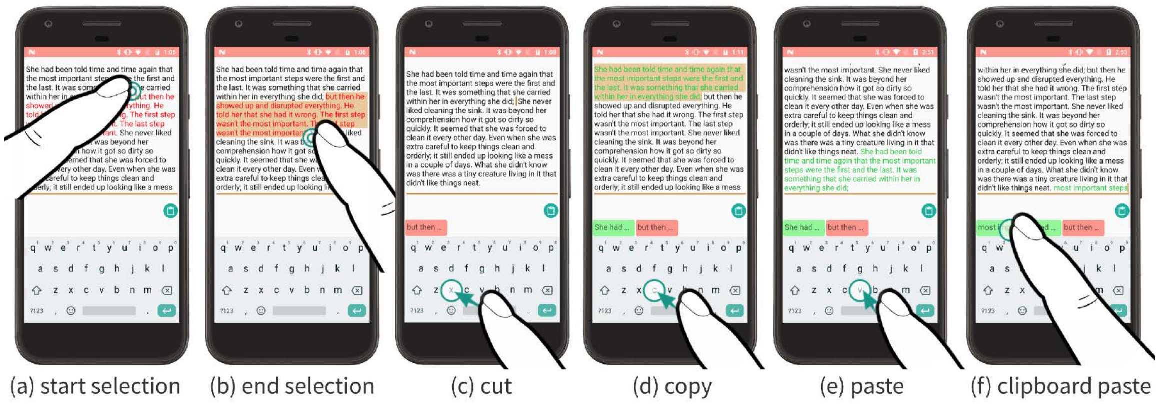


Figure 4.1: A cut–copy–paste workflow with GeShort: the user (a) positions the cursor and double-taps to initiate a selection, (b) positions the cursor and double-taps to complete the selection, (c) swipes from the Space to "X" to cut the selected text, (d) swipes from the Space to "C" to copy the selected text, (e) swipes from the Space to "V" to paste the selected text, (f) taps on a previously cut (highlighted in red background) or copied (highlighted in green background) text on the floating clipboard to paste it.

existing text to improve its quality. Text *formatting*, in the context of this work, represents styling existing text with bold, italicized, or underlined typeface.

Both text editing and formatting on mobile devices involve precisely positioning the cursor over the text to be selected, long-tap (500–1,000 ms [40]) to enable the "edit mode", which displays an edit toolbar and two handles to adjust the selection range (see Fig. 4.2b), adjust the selection range by dragging the handles, then select the intended task from the toolbar. If the intended task is not visible in the toolbar (which is usually the case for formatting tasks), users have to go to a secondary drop-menu by tapping on an icon, then select the option. Precise positioning of the cursor is difficult on smartphones due to the "fat-finger problem" [132]. Prior studies showed that users frequently make mistakes, requiring extra time for correction, when precisely positioning the cursor using touch [21]. The use of a dwell threshold (long-tap) and multiple menu selection actions also add to the time and complexity of text editing and formatting on mobile devices. Performing these actions are even more difficult when holding the device with one hand and interacting with the thumb of the same hand, which is one of the most common postures for mobile interaction [8]. Mobile users frequently use one hand to hold and enter text on mobile devices in situational impairments when the other hand is unavailable, such as when holding a coffee cup or a sandwich with one hand, performing dual tasks (e.g., navigating a desktop browser when texting, etc.), holding the hand of a toddler, or when relaxing on a couch. People who cannot use both hands due to a disability or amputation also use one hand to interact with mobile device. However, we do not include this population in this research.

Further, existing mobile text editing methods do not fully support delayed, repeated, and batch editing, thus not always possible to cut/copy text for use in a later editing episode (delayed pasting), cut/copy chunks of text for pasting in a preferred sequence (repeated pasting), or paste all cut/copied text with a single action (batch pasting). Instead, most methods require users to repeatedly perform the select-cut/copy-reposition-paste action sequence. Recently Gboard, the default Google keyboard [93], introduced a clipboard feature that stores the last five cut/copied text, enabling users to paste those later in a preferred sequence. While this feature supports delayed and repeated pasting (but not batch pasting), accessing the clipboard and locating its content is not straightforward, rather time-consuming and tedious, discussed in Section 4.3.

We propose GeShort, a method for one-handed text editing and formatting on mobile devices using <u>ge</u>stural <u>short</u>cuts. GeShort facilitates direct cursor positioning by using three simple rules, one-handed text editing and formatting with gestural shortcuts, which are inspired by the commonly used keyboard hotkeys, and delayed, repeated, and batch editing using a floating clipboard. The remainder of this Chapter is organized as follows. First, we review the existing works in the area and discuss the basic and advanced editing and formatting features on current mobile systems. We then present GeShort's text selection, editing, and formatting approaches. We evaluate GeShort in two user studies by comparing its basic and advanced features with the default Google keyboard's basic and advanced features. Finally, we conclude the work with reflections on future directions.

4.2 Related Work

There are not much works focused on text editing or formatting on mobile devices. Fuccella, Isokoski, and Martin [60] enabled text editing by performing one and two-finger gestures on the screen. In an evaluation, this method yielded 13–24% faster task completion time than Android OS 2.3's editing widget. In a follow-up work, Fuccella and Martin [61] used similar gestures to enable bimanual text editing. This method was 2% faster than Android OS 5.0's editing widget in a user study. Zhang and Wobbrock [142] used similar gestures to enable text editing with one and both hands. In a study, the one-hand approach yielded a 24% faster and the two-hand approach yielded a 17% faster task completion time than Android OS 9.0's editing widget. These methods, however, do not support text movements and use unfamiliar gestures that could be difficult to discover and learn [32]. In general, these works establish gestural interaction as an effective method for manipulating text on mobile devices. These works suggest that multi-finger gestures result in a faster task completion time than one-finger gestures but tend to increase physical efforts.

Ando et al. [7] developed a tap and tilt hybrid method with which users place the cursor at the beginning of the text, hold the "C" key and tilt the device to adjust the selection range, then release the key to copy the selected text. In a follow-up work, Ando et al. [6] replaced tilts with finger slides, where users use the keyboard as a trackpad to adjust the selection range. Both methods were reported to perform better than the Android OS 8.0's editing widget. But the actual performance of the methods are indeterminant due to the use of unconventional evaluation protocol and performance metrics. Besides, these methods do not support common editing tasks, such as cutting, pasting, or moving text. Roth and Turner [117] used the bezel of a device to enable text editing. With this method, users position the cursor at the beginning of the text, initiate a gesture from a specific area of the bezel to indicate cut or copy, then lift the finger at the end of the text to complete the corresponding task. In an evaluation, this method was 36% slower than the Apple iOS 2.0's editing widget. Chen et al. [40] adapted this method to enable cross-application copy-paste, where it yielded a 30% faster copying time than Android OS 4.1.2's editing widget. The study, however, used a mix of 15, 18, 21 sp sized fonts, two of which were larger than the recommended 16 sp [83]. Hence, it is unclear how the method would perform in real-world scenarios. Schweigert et al. [120] proposed using knuckle gestures for text editing tasks. Gutwin et al. [68], in contrast, augmented three push buttons on a smartphone case to enable text editing by performing chords. These methods were not evaluated in user studies. Alvina et al. [4] enabled text formatting by performing gestural shortcuts above the keyboard. Fennedy et al. [58] adapted actual keyboard hotkeys on a virtual keyboard, where

Table 4.1: Mobile text editing and formatting methods with their reported performance gains compared to the legacy editing and formatting features of smartphones. Notice that none of these methods support all commonly used editing tasks (i.e., copying, pasting, cutting, moving).

Reference	**Interaction**	**Editing**	**Formatting**	**Baseline**	**Gain**	
					Speed	**Accuracy**
Chen et al. [40]	Bezel gestures	Partial	No	Android OS	30%	−3%
Zhang and Wobbrock [142]	One-hand gestures	Partial	No	Android OS	24%	NA
Fuccella, Isokoski, and Martin [60]	One-hand gestures	Partial	No	Android OS	13-24%	NA
Zhang and Wobbrock [142]	Two-hand gestures	Partial	No	Android OS	17%	NA
Fuccella and Martin [61]	Two-hand gestures	Partial	No	Android OS	2%	NA
Roth and Turner [117]	Bezel gestures	Partial	No	iOS	−36%	0%
Ando et al. [7]	Tap and tilt	Partial	No	Android OS	NA	NA
Ando et al. [6]	Tap and swipe	Partial	No	Android OS	NA	NA
Gutwin et al. [68]	Two-hand chording	Partial	Yes	NA	NA	NA
Fennedy et al. [58]	Two-hand hotkeys	Partial	Yes	NA	NA	NA
Schweigert et al. [120]	Knuckle gestures	Partial	Yes	NA	NA	NA
Alvina et al. [4]	One-hand gestures	No	Yes	NA	NA	NA

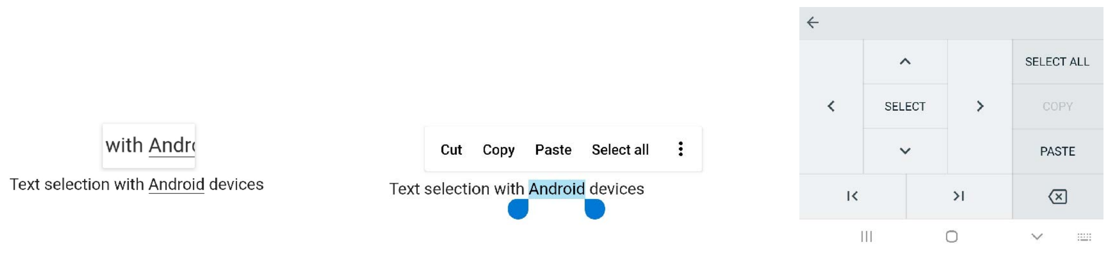
users tap on the keys in a sequence or with two thumbs. These methods were also not evaluated in user studies. Findings of these works suggest that exploiting device holding position and posture or external hardware and attachments do not result in a performance gain in short exposures. However, it is unclear whether these methods will improve performance with practice since neither of these were evaluated in longitudinal studies. Table 4.1 presents existing text editing and formatting methods for mobile devices and their performance gains reported in the literature.

Some have studied cursor positioning with mobile devices. Scheibel et al. [119] developed a virtual (joy)stick controller to control the movement of the cursor. Arif et al. [21] developed a smart cursor positioning system to facilitate error correction. Albanese et al. [2] proposed displaying directional arrows near the cursor to enable adjusting its position by tapping on them. These methods, however, were not compared with standard methods. In a different line of research, Zhao et al. [143] enabled text editing with speech commands. Eady and Girouard [56] developed a deformable prototype to demonstrate cursor control by bending the corners of the device. Le et al. [92] elicited (mostly) back-of-device gestures for the most important shortcuts for smartphones. Darbar et al. [49] used a smartphone as a trackpad, conduit for keyboard shortcuts, air-mouse, and ray casting device to enable cursor positioning in augmented and virtual reality. Pandey, Alizadeh, and Arif [111] developed a predictive system for number editing on smartphones. Some have also developed methods to facilitate error correction on mobile devices [86, 121, 21, 18, 9]. These, however, are outside the scope of this work.

4.3 Default Text Editing and Formatting Behaviors

Most mobile operating systems and third-party virtual keyboards use similar text editing features. Here, we focus mostly on the Google Android OS and its default keyboard, Gboard [93], behaviors.

Text Selection. Android enables users to directly position the cursor by touching the screen. The cursor is placed where the finger lands. Sliding the finger in any direction moves the cursor along with the finger within the text. A magnifier window appears to display the text under the finger to facilitate precise target selection (Fig. 4.2a). Newer Android devices enable users to place the finger on the Space and slide it left or right to move the cursor horizontally within the text [43]. Some Apple devices enable using the whole keyboard as a trackpad by long-tapping or applying extra force on the Space [106]. Long-tapping on text for 500–1,000 ms automatically selects the word under the finger and displays two text selection handles [40] (Fig. 4.2b). Users then can adjust the handles to


with Andr
Text selection with Android devices
Cut
Copy
Paste
Select all
Text selection with Android devices
SELECT ALL
SELECT
COPY
PASTE

(a) Magnifier window (b) Text selection handles and the edit menu (c) Gboard's text editing layout

Figure 4.2: Some cursor positioning, text selection, and editing features on Android-based devices.

modify text selection. Gboard also includes a dedicated text editing keyboard layout to enable users to move the cursor and select text with virtual keys (Fig. 4.2c).

Text Editing. Text editing on mobile devices is straightforward. Upon selection of text, a menu containing the options cut, copy, and paste appears above the text (Fig. 4.2b). Users then select an option to perform the corresponding task. Users could also use the dedicated keyboard layout for editing (see Fig. 4.2c) to perform these tasks. To reposition (or move) the text, users long-tap on selected text, then move the finger to the desired location.

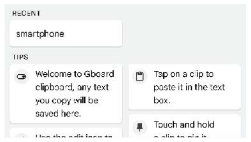
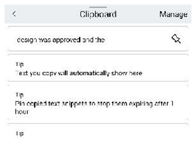
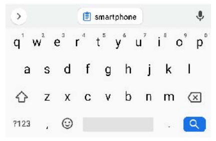
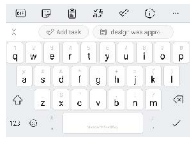
Text Formatting. Most mobile operating systems do not provide dedicated methods for text formatting, instead rely on the developers to include text formatting features in applications. Some applications display a text formatting tool above the virtual keyboard to bold, italicize, and underline the selected text (Fig. 4.3a). Web developers could also force Android to display some formatting options in the edit menu (Fig. 4.3b), accessed by tapping on the kebab menu icon (⋮).

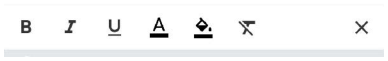
Default Clipboard. Gboard has recently included a clipboard feature to enable multiple cuts, copies, and pastes [118]. Once this feature is enabled from the settings, it displays the last cut or copied text in the suggestion bar. It also stores the last five cut and copied text for users to paste (Fig. 4.3c). This feature is particularly useful when users want to first cut or copy multiple chunks, then paste them when done (rather than repeated "cut/copy-paste" sequences). To access the recorded chunks, users expand the keyboard menu by pressing an arrowhead icon (<), then select the clipboard option. We reviewed the most popular virtual keyboards for smartphones, including Google Gboard, Apple iOS keyboard, Microsoft Swiftkey, Samsung keyboard, Fleksy, Grammarly, and Typewise, by using these on our devices for several days. It revealed that this feature is not supported by all keyboards. Table 4.2 presents an excerpt of the findings.

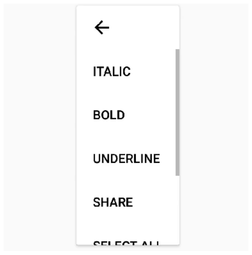
4.4 GeShort Text Editing and Formatting Behaviors

We collected data from a prior study [18], where participants were asked to transcribe text using a virtual keyboard. We also conducted a new pilot study with 12 participants (3 female, 9 male),

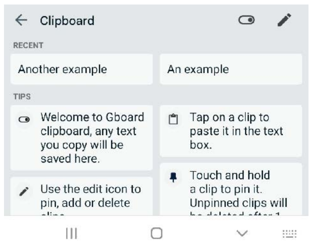
Table 4.2: Availability of advanced clipboard features in popular virtual keyboards. "NA" signifies not available.

Function	Gboard		Samsung	Microsoft SwiftKey	Default Apple iOS
	Android	Apple iOS			
Clipboard	 RECENT smartphone TIPS Welcome to Gboard clipboard, any text you copy will be saved here. Tap on a clip to paste it in the text box. Touch and hold a clip to pin it.	NA	NA	 Clipboard Manage design was approved and the Tip Text you copy will automatically show here Tip Pin copied text snippets to stop them expiring after 1 hour Tip	NA
Clipboard Buffer		NA	NA		NA


(a) Text formatting on Outlook for Android



(b) Android's text formatting options



Another example

An example

Welcome to Gboard clipboard, any text you copy will be saved here.

Tap on a clip to paste it in the text box.

Touch and hold a clip to pin it. Unpinned clips will

Use the edit icon to pin, add or delete

(c) Gboard's clipboard with all recorded text

Figure 4.3: Text formatting features in third-party applications and on Android-based devices, and the new clipboard feature of Gboard.

M = 27.92 years (SD = 4.87), where participants selected random chunks of text on a smartphone. After the pilot study, participants discussed their mobile text editing and formatting strategies and challenges. Based on the cursor positioning patterns identified in the studies, GeShort uses simple cursor positioning rules, and gesture-based cursor movement, text selection, editing, and formatting options, as well as a floating clipboard to enable faster and more accurate text editing and formatting on mobile devices.

4.4.1 GeShort Text Selection

We developed three simple rules to facilitate precise target selection.

1. This rule is based on the observation that users usually cut or copy from the beginning of one word to the end of the same or a different word. When the finger lands within three characters from the beginning or the end of a word (see Section 4.4.2), the cursor is placed at the beginning or the end of the respective word. When both the beginning of one word and the end of another word are within the three-character threshold:

   1.1 The cursor is placed at the beginning of the respective word if it is the first cursor placement for text selection.

   1.2 The cursor is placed at the end of the respective word if it is the last cursor placement for text selection.

2. This rule is based on the observation that for words with prefixes or suffixes, users tend to cut, copy, or delete the prefix or suffix. If the finger lands closer to the middle of a word that is either a gerund (words with "-ing"), compound (blends of multiple words, like "guessability" is a blend of "guess" and "ability"), plural form of a word (like "boxes"), or contraction (like "couldn't"), the cursor is placed between the prefix and the suffix. For example, if the finger lands closer to the middle of the word "guessability", the cursor is placed between "guess" (prefix) and "ability" (suffix). Our experimental prototype uses a dictionary to identify these words.

3. In all other cases, the method behaves like the default cursor positing method, described in Section 4.3.

GeShort enables users to move the cursor by one character at a time by swiping left and right (horizontal movement), and one line at a time by swiping up and down (vertical movement) on the text area. For text selection, users first position the cursor at the beginning of the intended text and double-tap to initiate selection. Users then re-position the cursor at the end of the text and double-tap to confirm the selection (Fig. 4.1). With this approach, users have to precisely position the cursor to indicate the start and the end of a selection, but the double-taps to confirm the selection do not have to be precise, rather tapping closer to the cursor is sufficient (within 20 pixels). We acknowledge that some mobile applications use double-taps to enable the selection of a whole word or a chunk of text, and to trigger the zoom-in feature on a document or website. We resolve the first conflict by enabling the selection of a whole word or chunk of text by long-tapping (500 ms) on the text. The second conflict can be avoided by tapping on the gutter or unused areas of a document or a website (i.e., not on text). To cancel or re-start a selection process, users double-tap on the cursor again. Algorithm 1 describes GeShort's text selection procedure.

Algorithm 1: GeShort Cursor Positioning

```
Input: Touch position $t$
Function CursorPlacement($t$):
    $c_{\text{default}} \leftarrow$ default cursor position closest to the touch $t$
    if $c_{default}$ points to whitespace then
        return $c_{\text{default}}$
    end
    $w \leftarrow$ the word that contains cursor $c_{\text{default}}$
    $s_w, e_w \leftarrow$ beginning and end cursor coordinates of word $w$
    if pixels$(c_{default} - s_w)$ < 20 then
        return $s_w$
    end
    else if pixels$(e_w - c_{default})$ < 20 then
        return $e_w$
    end
    else if $w$ is gerund then
        return $e_w - 3$
    end
    else if $w$ is plural then
        if $w$ ends with 's' then
            $e_w - 1$
        end
        else if $w$ ends with 'es' then
            $e_w - 2$
        end
    end
    else if $w$ is portmanteau or contraction then
        $s \leftarrow$ split point of the words $w$
        return $s$
    end
    return $c_{\text{default}}$
```

4.4.2 Three-Character Threshold

The three character threshold in Rule (1) is picked based on the mean touch contact area of the thumb (*w:* 6 × *h:* 7.2 mm) and the index finger (*w:* 5.5 × *h:* 6.7 mm) [134], which occludes about three characters in the default font type and size on most smartphones. Android OS, for instance, use the font Roboto at 18 sp, which takes on average *w:* 2.3 × *h:* 3.0 mm screen per character.

Since this threshold and the suffix-prefix cursor positioning in Rule (2) seldom forces users to swipe left or right to move the cursor to the right position (when an incorrect position is selected), we conducted a keystroke-level analysis to find out whether correction efforts outweigh their benefits. Correcting cursor position with GeShort can be described in the following keystroke-level model $t_P + (n \times t_K)$, where t_P is the time (ms) to position the cursor with two taps, t_K is the time to perform a swipe gesture, and n represents character offset, which is the total number of characters between the current and the target position. The model does not account for homing time (t_H, the time to move the finger to the target) and the mental preparation time (t_M, the time to visually scan the display, and prepare for the next task) for simplicity as these parameters are difficult to gather and separate from other parameters in user studies. Since these parameters are likely to be comparable between the methods, the model is still capable to estimate performance differences (%) between them.

We conducted a pilot study to collect the parameter values. Six participants (M =26.7 years, SD = 6.1) took part in the pilot study. Four of them identified as female and two as male. They were all experienced smartphone users (M = 8.33 years of experience, SD = 2.3). In the study, they held a smartphone with their dominant hand, then performed tap and the two swipe gestures using the thumb of the same hand. In order to collect direct cursor positioning time with the default method, they also precisely positioned the cursor at randomly selected positions in a paragraph using the same holding posture. Each task were performed 12 time, with ∼5 seconds breaks in between (6 × 12 × 4 = 288 data points). The study identified the following parameter values: $t_P = 320$, $t_K = 600$. Precise cursor positioning with the default method took on average 2,166 ms (SE = 356) including correction efforts (53% of all attempts required repositioning the cursor to the correct position). We then predicted cursor positioning and adjustment time with GeShort with $n = 0$ to 3. We did not consider an offset over three characters ($n > 3$) since the average word length in the English language is about 5 characters [109, 28], which, combined with Rules (1) and (2), assures that the offset will almost never be over three characters. Table 4.3 presents the predicted positioning and adjustment time with GeShort, where one can see that cursor re-positioning with three character offset is still 2% lower than the average 2,166 ms cursor positioning time with the default method. This further motivates the work.

4.4.3 GeShort Text Editing and Formatting

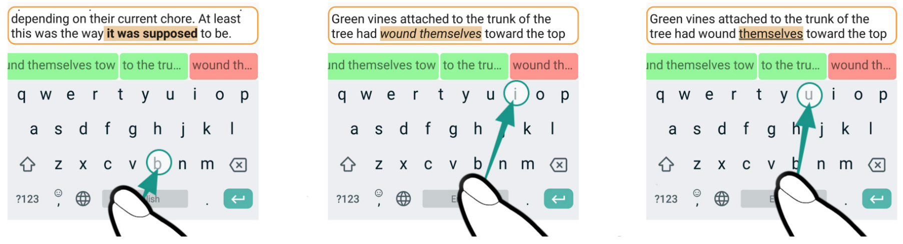
GeShort uses gestural shortcuts, designed based on keyboard hotkeys, for text editing and formatting. Table 4.4 presents these shortcuts and their desktop counterparts. These shortcuts replace the "Ctrl" or "Cmd" of keyboard shortcuts with a gesture initiated from the Space. For example, to copy a selected chunk of text, users swipe from the Space to the "C" key. Since these gestural shortcuts are initiated from the Space (Figs. 4.1, 4.5), they do not interfere with gesture-based text entry techniques like ShapeWriter or Swipe [141, 113]. Users can undo the last performed task by performing the undo shortcut (Space→Z).

Table 4.3: Predicted cursor positioning and adjustment time with GeShort, together with estimated performance gain in relevance to the average cursor positioning time with the default method.

Character Offset	**GeShort** $t_P + (n \times t_K)$	**Performance Gain**
$n = 0$	320 ms	85%
$n = 1$	920 ms	58%
$n = 2$	1,520 ms	30%
$n = 3$	2,120 ms	2%

Table 4.4: Proposed gestural shortcuts for text editing and formatting and their desktop counterparts. A '$\rightarrow$' signifies a gesture.

Shortcuts	**Cut**	**Copy**	**Paste**	**Bold**	**Italic**	**Underline**	**Undo**	**Redo**
Keyboard	Ctrl + X	Ctrl + C	Ctrl + V	Ctrl + B	Ctrl + I	Ctrl + U	Ctrl + Z	Ctrl + Y
Gestures	Space$\rightarrow$X	Space$\rightarrow$C	Space$\rightarrow$V	Space$\rightarrow$B	Space$\rightarrow$I	Space$\rightarrow$U	Space$\rightarrow$Z	Space$\rightarrow$Y


depending on their current chore. At least this was the way **it was supposed** to be.

Green vines attached to the trunk of the tree had *wound themselves* toward the top

Green vines attached to the trunk of the tree had wound <u>themselves</u> toward the top

(a) Bold (b) Italicize (c) Underline

Figure 4.4: Gestural shortcuts to (a) bold, (b) italicize, and (c) underline text, respectively. The shortcuts are initiated from the Space, thus do not interfere with gesture-based text entry techniques like ShapeWriter or Swipe.

4.4.4 GeShort Move Function

GeShort enables users to move text to a desired position, a feature not supported by most alternative approaches. Upon selection of a chunk of text, users can convert the virtual keyboard to a trackpad by holding the finger on the Space. When enabled, the keyboard is greyed out with all keys disabled. Users then move the finger over the trackpad to reposition the cursor to the intended position, and lift the finger to move the selected text to the cursor position. The design of this function is inspired by the drag and drop feature on desktop platforms, which enables users to "grab" an object to "drag" it to a different location. This is because, research [27] showed that exploiting specific knowledge that users already have of other domains (i.e., (re)using metaphors) facilitates learning.

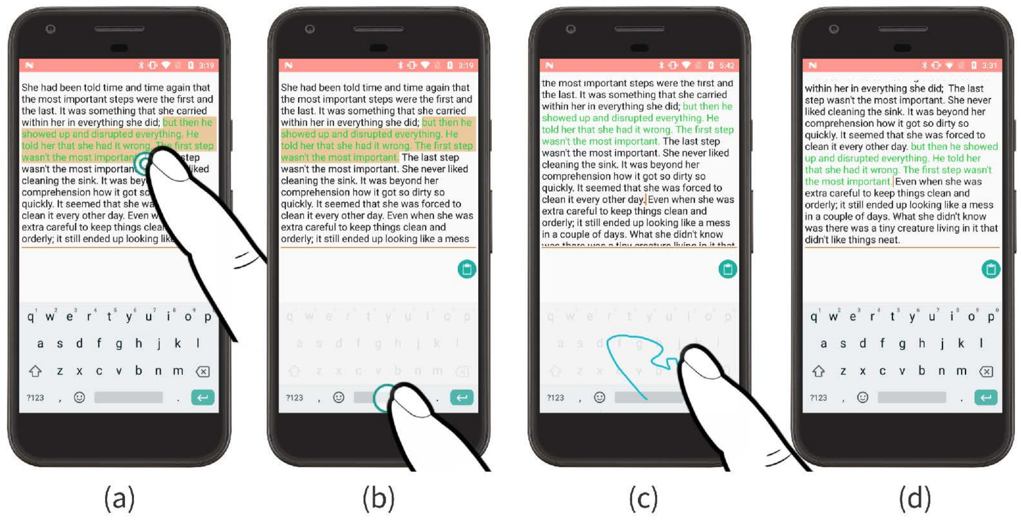


Figure 4.5: Moving text with GeShort: the user (a) selects text, (b) long-taps on the Space to turn the keyboard into a trackpad, (c) moves the finger over the trackpad to reposition the cursor, (d) lifts the finger to place the selected text to the cursor position.

4.4.5 GeShort Floating Clipboard

GeShort provides easy access to the clipboard to facilitate delayed, repeated, and batch editing tasks. Unlike the default clipboard, which users have to access by extending the keyboard options, GeShort automatically displays a clipboard icon when it has cut or copied text (Fig. 4.6). Users tap on the icon to see snippets (i.e., the first nine characters of the text) of the content in a bar, then tap on a snippet to paste the corresponding text. Once opened, the clipboard bar remains visible until users tap on the icon again to hide it. The bar distinguishes cut and copied text using red and green backgrounds, respectively. This is based on prior research that showed that distinguishing different types of text editing actions using different colors improves usability and performance [3, 20]. Users can swipe left and right on the bar to access cut and copied text that are not visible in the bar. Users can extend a snippet by long-tapping on it. The text shrinks back to preview size upon lifting the finger to accommodate more excerpts in the bar. GeShort supports the following gestural shortcuts to afford users better control of its behavior, which are not supported by the default clipboard: Space→"V" pastes the last cut or copied text, Space→Enter pastes all items from the clipboard (to enable batch pasting with one action), and Space→Delete clears the clipboard.

4.5 User Study 1: Default vs. GeShort

We conducted a user study to compare the basic editing (cut, copy, paste, move, and delete) and formatting (bold, italic, underline) tasks with the default Android keyboard Gboard and GeShort.

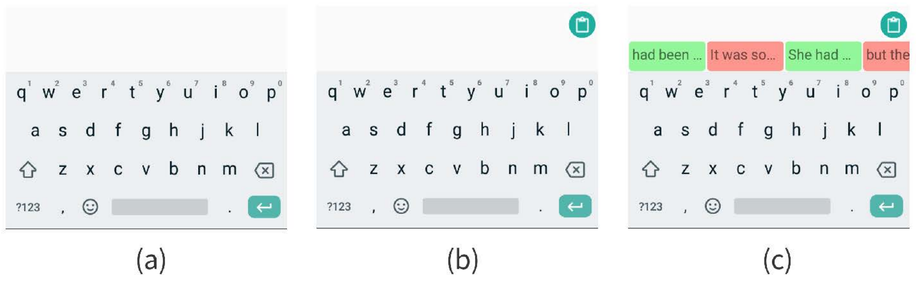


Figure 4.6: Different states of GeShort's floating clipboard: (a) the clipboard icon is not visible when it is empty, (b) the clipboard icon appears when it has content, (c) tapping on the icon displays recently cut and copied text in a bar above the keyboard, highlighted in red and green backgrounds, respectively.

4.5.1 Apparatus

We used a Motorola Moto G^5 Plus smartphone ($150.2 \times 74 \times 7.7$ mm, 155 g) at 1080×1920 pixels running on Android OS 7.0 in the user study. We developed a custom application with Android Studio 3.5.1, SDK 24 to display the tasks and record all interactions with timestamps. The floating clipboard was disabled in this study since only basic editing tasks were investigated. The application had four parts: a paragraph at the top, followed by a section presenting one task at a time, a text area to paste the cut/copied text, and a virtual keyboard (Fig. 4.7).

4.5.2 Participants

Twelve participants aged from 20 to 32 years (M = 24.83, SD = 4.13) took part in the study. They were recruited through social networking platforms, e-mailing lists, and word of mouth. Eight of them identified themselves as female and four as male. All of them were experienced mobile users (M = 8.04 years of experience, SD = 3.84). Ten of them were owners of Apple iOS-based smartphones, on which they used either the default Apple keyboard (N = 8) or Gboard (N = 2). The remaining two were Android OS-based smartphone owners, on which they used either the default Samsung keyboard (N = 1) or Gboard (N = 1). They all were proficient in the English language (native, bilingual, or advanced-level, moderate speakers of the language). All of them were right-handed. They all received U.S. $15 for participating in the study. Fig. 4.7 illustrates the custom applications and two volunteers participating in the study.

4.5.3 Design

The study used a within-subjects design with two independent variables: 1) *method* with two levels: default Gboard and GeShort and 2) *task* with two levels: editing and formatting. The design was as follows:

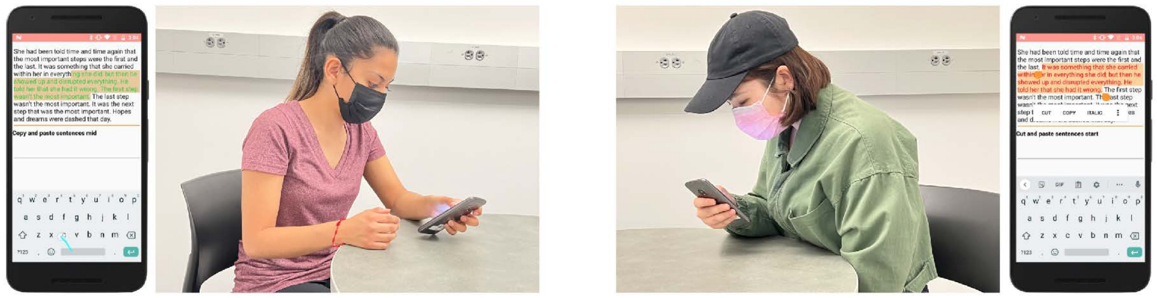


Figure 4.7: The device with GeShort (left) and the default method (right), and two volunteers participating in the study.

12 participants ×
2 methods (default, GeShort), counterbalanced ×

counterbalanced
randomized

4 × 16 editing tasks (cut, copy, move, delete)
= 1,536 editing tasks, in total

3× 16 formatting tasks (bold, italic, underline)
= 1,152 formatting tasks, in total

4.5.3.1 Task Selection

We carefully selected the experimental tasks to increase the external validity of the study. In the editing phase, participants performed sixteen cut, copy, move, and delete tasks with each method in randomized order. While in the formatting phase, they performed sixteen bold, italic, and underline tasks with each method in randomized order. The text to be edited or formatted were equally split between word-level and phrase-level selection tasks, the former composed of 1–3 words and the latter composed of multiple lines of text. Word-level and phrase-level selection tasks were further divided into mid-selection and start/end-selection tasks, the former required selecting text within a word and the latter required selecting text from/to the start/end of a word. For the tasks, we generated eight random paragraphs using a freely available service[1]. We used random paragraphs to reduce the effects of the content and the context of the paragraphs on the selection tasks. Using excepts from existing sources could have introduced a confound since participants' familiarity to the text cannot be predicated ahead of the study. Table 4.5 presents four examples of these tasks. Table 4.6 presents statistics about the paragraphs.

4.5.3.2 Performance Metrics

The dependent variables were the following performance metrics.

- **Task completion time (s)** is the average time (in seconds) participants took to complete one task. For deeper analysis, we divided task completion time into selection time and action time, representing the average time to select text and the average time to perform an editing or formatting task, respectively. Hence, *task completion time* = *selection time* + *action time*.

[1]Random Paragraph Generator: https://randomwordgenerator.com/paragraph.php.

Table 4.5: Examples of experimental tasks of different selection patterns.

Selection Level	Task	Example
Word-level start/end-selection	Copy	**Copy and Paste words beginning** She had been told time and time again that the most important steps were the first and
Word-level mid-selection	Cut	**Cut and Paste words middle** was captivating by reason of a certain frankness of expression and a contradictory
Phrase-level start/end-selection	Move	**Move sentences beginning** could see movement. She squinted her eyes and peered in the direction of the movement; trying to decipher exactly what she had spied.
Phrase-level mid-selection	Bold	**Bold sentences middle** happened. It didn't even really make sense to him. All he knew was that he froze at the moment; and nothing in his body allowed to move. It was as if he had instantly become

- **Errors per task** represents the average character errors committed per selection task. This metric is comparable to the minimum string distance measure in text entry research that represents the similarity between two text sequences using the Levenshtein distance algorithm [95]. The distance is defined as the minimum number of primitive operations (namely insertion, deletion, and substitution) needed to transform a cut or copied text to the text presented in the task [122].

4.5.4 Procedure

The study was conducted in a quiet room, one participant at a time. Upon arrival, we explained the research and collected their informed consent forms and demographics. Participants sat in front of a desk. They were asked to hold the device with the dominant hand in a comfortable position and interact with the thumb of the same hand. We then demonstrated the method used in the first condi-

Table 4.6: Statistics about the randomly generated paragraphs used in the studies.

Paragraph ID	Total Character	Total Word	Total Line
1	419	78	10
2	445	84	11
3	498	92	12
4	519	88	13
5	398	80	10
6	425	83	11
7	464	89	13
8	496	90	12
Mean	458	86	12

tion (the conditions were counterbalanced) and asked them to practice all features by performing at least two tasks per feature. They could extend the duration of the practice by one additional task per feature upon request. We then started the first condition, where they were asked to perform the tasks as fast and accurate as possible. The custom application displayed one task at a time. In addition to written instructions, the relevant parts of the paragraphs were highlighted in different colored font to reduce the visual scan time: green for copy and move, red for cut and delete. Formatting tasks were highlighted in a purple-colored font (Table 4.5). After completing a task, participants tapped on the Next key to see the next task. Correcting selection errors was not required. After they completed all tasks in the condition, we demonstrated the next method, asked them to practice with it, then started the next condition. We asked participants to practice with both methods, although they were familiar with Gboard, to make sure they know how to use the default editing features of the keyboard. In the practice, participants used all features by performing at least two tasks per feature (see Section 4.5.3.1). Participants were instructed to read the presented task carefully before initiating a task. The tasks used in the training were not repeated in the main study.

During the study, we did not restrict participants from using any default features, including the clipboard or cursor movements by swiping on the space key. Yet, participants almost never used these features. We enforced a 2-minute break after each condition to avoid the effect of fatigue. Participants could extend the duration of the breaks by 2 extra minutes upon request. After the study, participants completed the NASA-TLX questionnaire to rate the methods' perceived workload [73] and a custom questionnaire to rate the perceived speed, accuracy, ease-of-use, integration of the functions, and willingness-to-use on a 5-point Likert scale. For analysis, we calculated the raw TLX scores by individual sub-scales, which is a common practice in the literature [72].

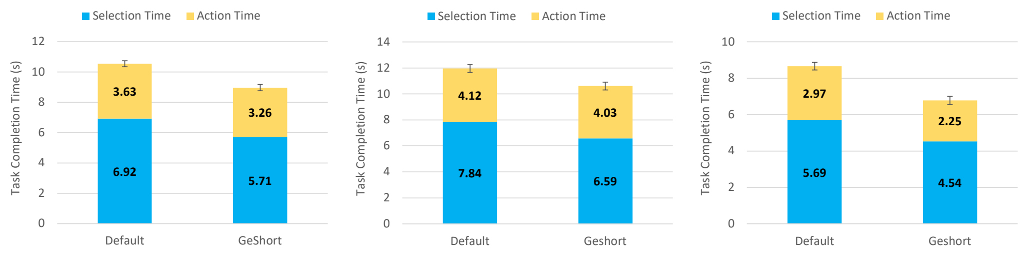
All researchers involved in this study were fully vaccinated for COVID-19. All participants were pre-screened for COVID-19 symptoms during the recruitment process by a researcher, and on the day of the study by the host institute. Both the researcher and the participants wore face coverings and sanitized their hands before a session. The researcher also maintained a 3-feet distance from the participants at all times. All devices and surfaces were disinfected before and after each session. This protocol was reviewed and approved by the UC Merced Institutional Review Board (IRB).

4.5.5 Results

A complete study session took about 60 minutes, including demonstration, questionnaires, and breaks. A Shapiro-Wilk test revealed that the response variable residuals were normally distributed. A Mauchly's test indicated that the variances of populations were equal. We did not exclude any outliers from the analysis. We used a repeated-measures ANOVA for the quantitative within-subjects factors and a Wilcoxon Signed-Rank test for the questionnaire data.

4.5.5.1 Task Completion Time

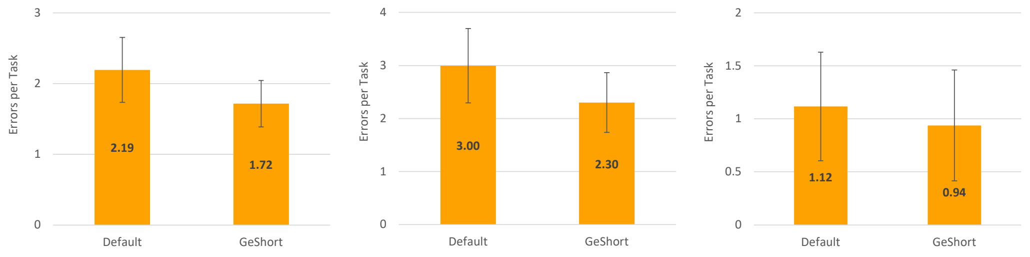
An ANOVA identified a significant effect of method on task completion time ($F_{1,11} = 17.29, p < .005$). On average, the default and GeShort took 10.56 s (SE = 0.20) and 8.99 s (SE = 0.20) to complete a task, respectively. The effects on selection time ($F_{1,11} = 11.00, p < .01$) and action time ($F_{1,11} = 13.87, p < .005$) were also statistically significant. The 5.7 s and 6.9 s selection time values are higher than the predicted values in Table 4.3 as the latter did not account for the homing or the mental preparation time. The 17% performance gain identified here falls between the gains predicted



(a) Overall (b) Editing (c) Formatting

Figure 4.8: The average overall, editing, and formatting task completion time with the two methods segmented in selection and action time. Error bars represent ± 1 standard error (SE).

for 2–3 character offsets. We further analyzed the data after filtering for editing and formatting tasks, where an ANOVA identified significant effects of method on both editing ($F_{1,11} = 5.44, p < .05$) and formatting ($F_{1,11} = 76.97, p < .0001$) task completion time. Fig. 4.8 illustrates the average overall, editing, and formatting task completion time split into selection and action time.



(a) Overall (b) Editing (c) Formatting

Figure 4.9: The average overall, editing, and formatting errors per task with the two methods. Error bars represent ± 1 standard error (SE).

4.5.5.2 Errors per Task

An ANOVA failed to identify a significant effect of method on errors per task ($F_{1,11} = 0.39, p = .54$). On average the default and GeShort caused 2.19 (SE = 0.46) and 1.72 (SE = 0.33) errors per task, respectively. We also failed to identify significant effects when filtered the data for editing ($F_{1,11} = 0.44, p = .52$) and formatting ($F_{1,11} = 0.04, p = .84$) tasks. Fig. 4.9 illustrates the average overall, editing, and formatting errors per task.

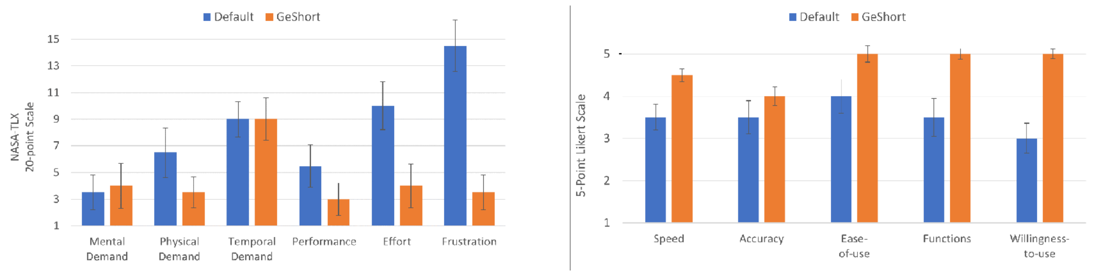
4.5.5.3 Perceived Workload

A Wilcoxon Signed-Rank test identified a significant effect of method on physical demand ($z = -2.14, p < .05$), effort ($z = -2.52, p <= .05$), and frustration ($z = -2.95, p < .005$). However, no significant effect was identified on mental demand ($z = -0.28, p = .78$), temporal demand

($z = -0.53, p = .69$), or performance ($z = -0.71, p = .48$). Fig. 4.10a illustrates median raw NASA-TLX ratings of both methods.

4.5.5.4 Usability

A Wilcoxon Signed-Rank test identified a significant effect of method on perceived speed ($z = -2.71, p < .01$), ease-of-use ($z = -2.39, p < .05$), function ($z = -2.38, p < .05$), and willingness-to-use ($z = -2.7, p < .01$). However, no significant effect was identified on perceived accuracy ($z = -1.85, p = .06$). Fig. 4.10b illustrates median user ratings of both methods.



(a) Raw NASA-TLX (b) Usability

Figure 4.10: Median user ratings of both methods on perceived workload and usability questionnaires. Error bars represent ±1 standard error (SE).

4.5.6 Discussion

Participants were significantly faster with GeShort than Gboard (15% faster). This performance gain cannot be directly compared with the performance reported in the literature (Table 4.1) since those works were evaluated with much simpler tasks (did not consider all selection scenarios described in Section 4.5.3.1), compared only a subset of edit options (did not compare cut or move), or did not support one-handed interactions. An analysis revealed that participants performed both text selection and editing/formatting tasks significantly faster than the default method (11% and 17% faster, respectively), which suggests that both the proposed selection and editing/formatting approaches contributed to the faster task completion time of GeShort. When filtered the data for editing and formatting tasks, we found out that both types of tasks were performed significantly faster with GeShort than the default method (11% faster editing and 22% faster formatting). Error rates of the two methods were not statistically significant.

The results of subjective feedback are also encouraging. In the perceived workload questionnaire (Fig. 4.10a), participants found GeShort to be significantly less physically demanding, which suggests that it better facilitates text editing and formatting with one hand than the default method. One participant (female, 20 years) commented that the default method *"was more physically and mentally challenging"* because its selection and editing/formatting features were not suited for one-handed interaction. Another participant (female, 31 years) praised GeShort saying that it *"was more precise in terms of text selection."* They also felt that the default method required significantly more effort, causing significantly more frustration performing the tasks. One participant (female, 26 years) commented, *"frustration occurred mostly when selecting with Google method."* Participants found both methods somewhat comparable in terms of metal and temporal demands.

In the usability questionnaire (Fig. 4.10b), participants found GeShort to be significantly faster and easier to use than the default method. One participant (female, 20 years) commented, *"The underline, bold, italic in GeShort was awesome! As someone who edits papers and writes essays on my phone I would actually use underline and such more often."* They felt that various functions were much better integrated in GeShort than the default method, making it substantially easier to use than the baseline. One participant (female, 26 years) commented that *"[various editing] functions [are] a lot easier to use with GeShort."* As a result, participants preferred using GeShort on their mobile devices significantly more than the default method.

4.6 User Study 2: Default vs. GeShort Clipboard

We conducted a second user study to compare the advanced clipboard features (delayed and repeated pasting from the clipboard) of the default Android keyboard Gboard and GeShort.

4.6.1 Apparatus & Participants

The study used the same apparatus as the first study (Section 4.5.1). Twelve new participants took part in this study. They were recruited through social networking platforms, e-mailing lists, and word of mouth. Their age ranged from 18 to 31 years (M = 22.30, SD = 3.42). Six of them identified themselves as female and six as male. All of them were experienced mobile users (M = 9.46 years of experience, SD = 4.26). They all were owners of Apple iOS-based smartphones, on which they used the default Apple keyboard. All of them were proficient in the English language (native, bilingual, or advanced-level speakers of the language). Eleven of them were right-handed mobile users, while one was left-handed. They all received U.S. $15 for participating in the study. Fig. 4.11 illustrates the custom application and two volunteers participating in the study.

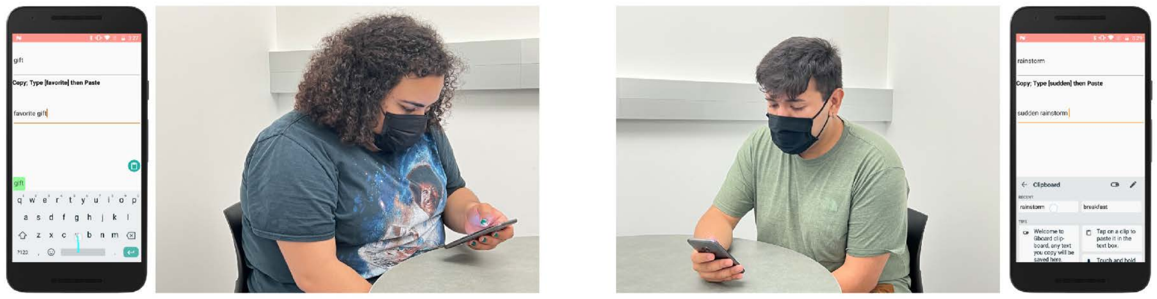

gift
Copy; Type [favorite] then Paste
favorite gift
rainstorm
Copy; Type [sudden] then Paste
sudden rainstorm
Clipboard
breakfast

Figure 4.11: The experimental device with GeShort and the floating clipboard (left) and the default clipboard (right), and two volunteers using the respective methods in the second user study.

4.6.2 Design & Procedure

The study used a within-subjects design with one independent variable *method* with two levels: default Gboard and GeShort. The design was as follows:

12 participants $\times$
2 methods (default, GeShort), counterbalanced $\times$ 20 tasks
$=$ 480 clipboard tasks in total.

4.6.2.1 Task Selection

Like the first user study, we carefully selected the experimental tasks to increase the external validity of the study. In the study, participants performed twenty clipboard tasks with each method in randomized order. Four different types of tasks were selected: 1) cut/copy, then paste from the suggestion bar, 2) cut/copy, type a word, then paste from the clipboard, 3) paste existing entries from the clipboard, 4) paste the same text 2–5 times (repeated paste). These tasks were selected to replicate real-life scenarios. For example, assume a user copied an address to share with a friend via text message. She could either paste it immediately (task 1), paste it after typing a message like *"here's the address"* (task 2), share the address later from the clipboard (task 3), or share it with multiple friends (task 4). To avoid potential confounding factors, the experimental tasks involved only one word that users could select by long-tapping on it (to eliminate the need for precise target selection), and pre-populated the clipboard with five words (to reduce the effect of visual scan time). Table 4.7 presents examples of these tasks.

4.6.2.2 Performance Metrics

The dependent variables were the task completion time and the errors per task performance metrics described in Section 4.5.3.2. However, unlike the first study, we did not split the task completion time by actions as performing these tasks required performing different numbers and combinations of actions.

4.6.2.3 Clipboard-specific Analysis

In the study, only the treatment group was allowed to use the gestural shortcuts, while the control group used the default selection methods. Hence, the above results present the combined benefits of the clipboard and the gestural shortcuts. We conducted a deeper analysis to compare only the performance of the two clipboard methods. For this, we filtered the data for the tasks that do not require using the shortcuts for text selection and editing, namely tasks 3 and 4 (50% of the total data). An ANOVA identified a significant effect of clipboard method on task completion time ($F_{1,11} = 54.65, p < .0001$). On average the default and the floating clipboard took 4.64 s (SE = 0.2) and 2.00 s (SE = 0.13) to complete a task, respectively. An ANOVA failed to identify a significant effect of clipboard method on errors per task ($F_{1,11} = 0.00, p = .95$). Both clipboard methods caused about 0.2 errors per task.

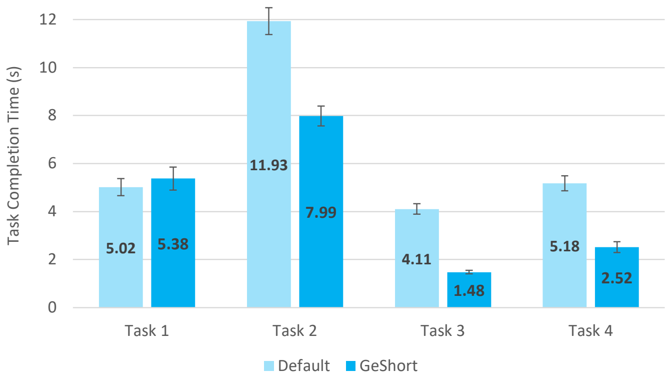


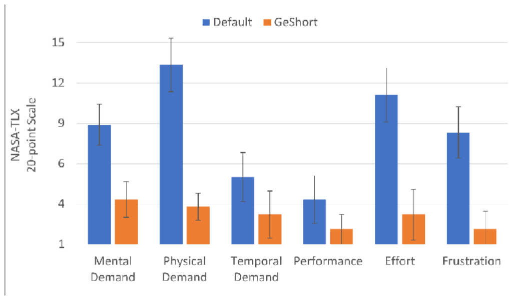
Figure 4.12: Average task completion time (s) per clipboard task. Error bars represent ± 1 standard error (SE)

4.6.2.4 Perceived Workload

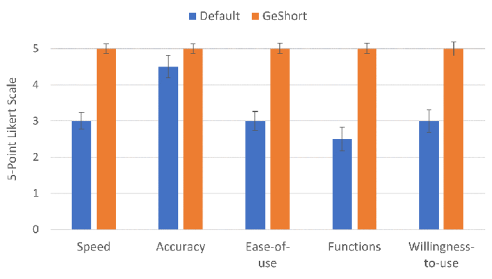
A Wilcoxon Signed-Rank test identified a significant effect of method on mental demand ($z = -2.20, p < .05$), physical demand ($z = -2.81, p < .01$), performance ($z = -2.67, p < .01$), effort ($z = -2.40, p <= .05$), and frustration ($z = -2.61, p < .01$). However, no significant effect was identified on temporal demand ($z = -0.93, p = .35$). Fig. 4.13a illustrates median raw NASA-TLX ratings of both methods.

4.6.2.5 Usability

A Wilcoxon Signed-Rank test identified a significant effect of method on perceived speed ($z = -3.13, p < .005$), ease-of-use ($z = -2.99, p < .005$), function ($z = -2.74, p < .01$), and willingness-to-use ($z = -2.65, p < .01$). However, no significant effect was identified on perceived accuracy ($z = -1.93, p = .05$). Fig. 4.13b illustrates median user ratings of both methods.



(a) Raw NASA-TLX



(b) Usability

Figure 4.13: Median user ratings of both methods on perceived workload and usability questionnaires. Error bars represent ± 1 standard error (SE).

4.6.3 Discussion

Participants were significantly faster with GeShort than the default method (34% faster). They were also significantly faster when using only the clipboard features (35% faster). A deeper investigation revealed that participants were significantly faster in performing in-text pasting (task 2), delayed pasting (task 3), and repeated pasting (task 4) with GeShort than the default method (33%, 64%, and 51% faster, respectively), while performance of immediate pasting tasks (task 1) was relatively comparable between the methods (Fig. 4.12). Error rates of the two methods were not statistically significant. These results suggest that GeShort better facilitates performing advanced editing tasks than the most current features of the state-of-the-art.

Results of subjective feedback also support this. In the perceived workload questionnaire (Fig. 4.13a), participants found GeShort to be significantly less demanding both mentally and physically. They also found the default method significantly more demanding than GeShort in terms of effort. They felt that the reduced mental and physical demands and effort significantly improved their performance with GeShort. One participant (female, 22 years) summarized, *"It took longer [with the default method] than anticipated! I never realized how many steps I had to take to do copy and paste. Using the GeShort method was a lot of simpler than imagined. The more I used it, the easier it gets."*

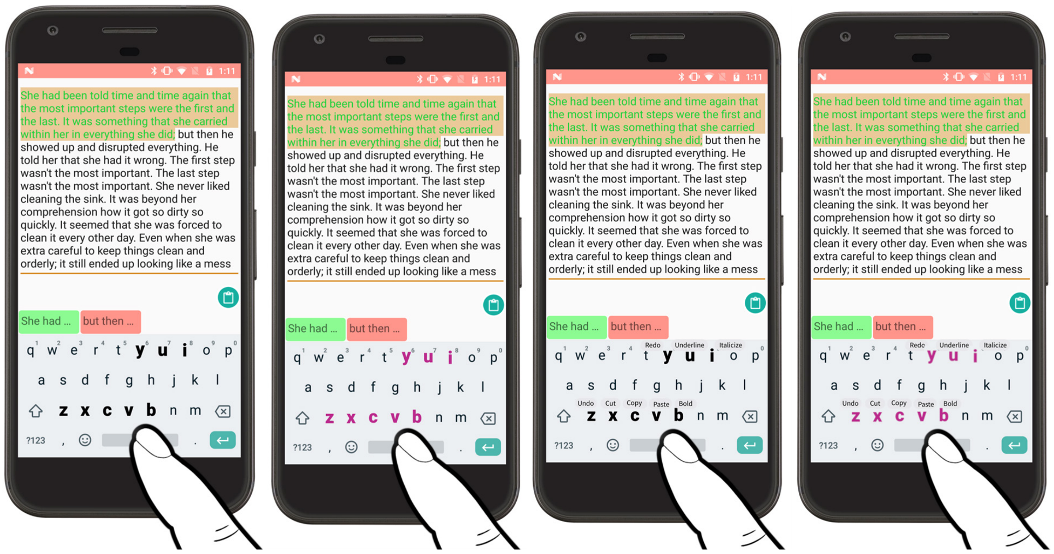
In the usability questionnaire (Fig. 4.13b), participants found GeShort to be significantly faster and easier to use than the default method. They felt that various functions were better integrated in GeShort than the default method, making it substantially easier to use than the baseline. Particularly, they appreciated how the gestural shortcuts matched the keyboard shortcuts. One participant (male, 21 years) commented that this made performing editing tasks *"much easier on GeShort."* Hence, participants preferred using GeShort on their mobile devices significantly more than the default method. One participant (female, 23 years) summarized, *"I found [the] new method easier to use and noticed I typed faster. GeShort made typing more accessible in my experiences especially when compared to Google."*

4.7 Conclusion

We presented GeShort, a text editing and formatting method for mobile devices. GeShort facilitates direct cursor positioning by using three simple rules, one-handed text editing and formatting with gestural shortcuts inspired by keyboard hotkeys, and delayed, repeated, and batch editing using a floating clipboard. We compared GeShort with the state-of-the-art Gboard in two user studies. Results of the first study revealed that the proposed method reduces text editing time by 11% and text formatting time by 22%. When the tasks are broken down into text selection and editing actions, GeShort reduces selection time by 11% and action time by 17%, validating the benefits of the method's selection approach and gestural shortcuts. Results of the second study revealed that the floating clipboard outperforms the clipboard feature of Gboard by 34% for advanced editing tasks. Besides, in both studies, participants found GeShort less onerous in terms of mental demand, physical demand, and effort, improving their overall performance with the method. They perceived GeShort as faster and easier to use than Gboard, felt that its functions were better integrated, and wanted to keep using it on their mobile devices.

4.7.1 Limitations

We acknowledge several limitations of the work. First, GeShort uses gestural shortcuts, which are designed based on keyboard hotkeys. Hence, those who are unfamiliar with the hotkeys may require extra time to learn the method. Limited discoverability of gestural interactions have long been a discussion in the literature [137]. Further investigations, particularly in-the-wild studies, are needed to find out whether users are able to discover and use the method in the real-world. Second, we used common editing and formatting tasks in the user studies, thus it is unclear whether the findings are generalizable to more complex editing and formatting tasks. Although it is doubtful that users would choose to perform such complex tasks on mobile devices. We encourage the research community to explore this further. Finally, we used convenience sampling for recruiting participants, which resulted in mostly tech-savvy young adults in the user studies. Therefore, the results reported here may not be generalizable to a more diverse population.


She had been told time and time again that the most important steps were the first and the last. It was something that she carried within her in everything she did; but then he showed up and disrupted everything. He told her that she had it wrong. The first step wasn't the most important. The last step wasn't the most important. She never liked cleaning the sink. It was beyond her comprehension how it got so dirty so quickly. It seemed that she was forced to clean it every other day. Even when she was extra careful to keep things clean and orderly; it still ended up looking like a mess

(a) Bold font (b) Colored font (c) Bold font & labels (d) Colored font & labels

Figure 4.14: Four key highlighting approaches will be explored in the future to facilitate gestural shortcut discovery and learning.

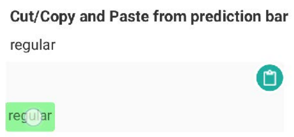
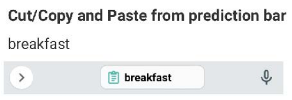
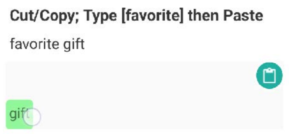
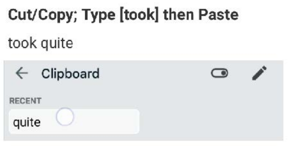
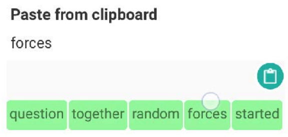
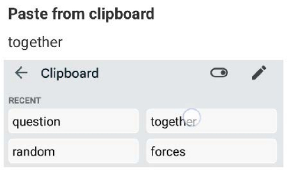
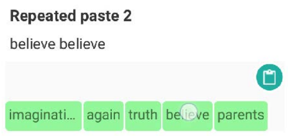
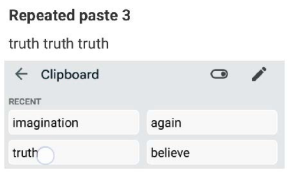
4.8 Future Work

In the future, we will develop a more sophisticated approach to facilitate text selection on mobile devices. For this, a longitudinal study will be conducted to understand users' text selection behaviors, then a machine learning model will be developed to predict potential selection actions. We will also explore the possibility of customizing the method for other touchscreen-based devices, particularly smartwatches and interactive tabletops.

4.8.1 Key Highlighting

The current interface does not provide any visual cues to facilitate the discovery and learning of the gestural shortcuts. In the future, we will investigate whether highlighting all possible shortcut keys as the user touches or initiates a gesture from the Space accommodates discovery or improves learning and performance since such approaches have been shown to improve input performance in the literature [62, 23, 66]. Particularly, four different highlighting approaches will be explored: 1) bold font, 2) colored font, 3) bold font with labels, 4) colored font with labels, as illustrated in Fig. 4.14.

Table 4.7: Examples of the four types of clipboard tasks used in the second study.

Task	GeShort	Gboard
Cut/Copy, then paste from suggestion bar	 **Cut/Copy and Paste from prediction bar** regular regular	 **Cut/Copy and Paste from prediction bar** breakfast breakfast
Cut/Copy, type, then paste from suggestion bar	 **Cut/Copy; Type [favorite] then Paste** favorite gift gift	 **Cut/Copy; Type [took] then Paste** took quite Clipboard RECENT quite
Paste existing entries from the clipboard	 **Paste from clipboard** forces question together random forces started	 **Paste from clipboard** together Clipboard RECENT question together random forces
Paste text multiple times (repeated paste)	 **Repeated paste 2** believe believe imaginati... again truth believe parents	 **Repeated paste 3** truth truth truth Clipboard RECENT imagination again truth believe

Chapter 5

Conclusion & Future Work

This dissertation investigates innovative methods for text and numeric input on mobile and wearable devices, aiming to enhance input speed, accuracy, and user satisfaction. By utilizing the current functionalities of smartphones and smartwatches, we seek to offer alternative input options that can significantly improve the user experience

Initially, we developed three novel methods for number selection on smartwatches: directional swipes, wrist twisting, and adjusting screen contact force. These methods offer a significant improvement over traditional number pickers by allowing users to smoothly switch between slow-and-steady and rapid continuous adjustments. We conducted two extensive user studies to assess these methods under various conditions, including both stationary and mobile use, as well as for selecting individual numbers and editing numbers within a text. The swipe-based method emerged as a standout, showing significantly quicker input speeds in all scenarios. Users praised its efficiency, precision, and low demand on mental and physical effort, with accuracy rates remaining high across different situations. This underscores the swipe method's potential as an optimal solution for smartwatch number selection.

Looking ahead, we plan to test the effectiveness of our number selection methods on larger touchscreen devices, like smartphones and tablets. Additionally, we aim to explore new selection techniques utilizing the unique hardware features of smartwatches, such as the crown and bezel, for further innovations in number picking.

Subsequently, we explored the use of contact force as a novel interaction method on smartwatches, specifically to improve selection accuracy for smaller targets. We conducted three user studies to examine this. First, we determined the comfortable range of force that users can apply to smartwatches. Second, we found that force input significantly outperforms traditional touch in selecting smaller targets. Third, we introduced and evaluated two new force-based methods for character-level text entry, showcasing their effectiveness in real-world scenarios. Beyond demonstrating the utility of force input, these keyboards are noteworthy contributions in their own right, offering more compact designs compared to existing character entry methods and facilitating quick user adaptation. These keyboards are envisioned as complementary tools for predictive keyboards, particularly in bypassing limitations like the prevention of out-of-vocabulary word entry due to aggressive autocorrection, thereby allowing for the inclusion of occasional non-dictionary words.

In the future, we aim to investigate different control-display mapping functions for force input and apply machine learning techniques to enhance reliability. Furthermore, we plan to conduct longitudinal studies on these new keyboards and integrate them with predictive text systems to achieve faster text entry.

Finally, we introduced GeShort, a novel method for text editing and formatting on mobile devices. GeShort revolutionizes the editing process by simplifying direct cursor placement with three easy-to-follow rules, enabling one-handed editing and formatting through gestural shortcuts inspired by keyboard commands, and enhancing delayed, repeated, and batch editing with a floating clipboard. We evaluated GeShort against the leading Gboard in two user studies. The first study's findings indicated that our method reduces text editing time by 11% and formatting time by 22%. Further analysis into text selection and editing actions showed an 11% reduction in selection time and a 17% decrease in action time, underscoring the efficacy of GeShort's selection technique and gestural shortcuts. The second study demonstrated that GeShort's floating clipboard feature surpasses Gboard's clipboard in efficiency by 34% for complex editing tasks. Additionally, participants reported that GeShort was less mentally and physically demanding and required less effort, which enhanced their overall experience. They found GeShort to be quicker and more user-friendly than Gboard, appreciated its integrated functionality, and expressed a desire to continue using it on their devices.

Looking ahead, we plan to refine text selection mechanisms on mobile devices further. A longitudinal study will be carried out to observe user behaviors around text selection, followed by the development of a machine learning model to predict likely selection actions. Moreover, we will investigate how to tailor this method for other touchscreen devices, especially smartwatches and interactive tabletops.

The results showcased in this dissertation demonstrate that numeric and character input and editing can be significantly enhanced in terms of speed, accuracy, and user preference. These improvements are realized through meticulous interface and interaction design, the adoption of straightforward rules, and the application of language models, leveraging the existing features of mobile and wearable devices. Crucially, achieving these advancements does not necessitate the introduction of additional sensors, new hardware, or the increased computational demands associated with sophisticated learning models.

Bibliography

[1] Youngseok Ahn et al. “BandSense: Pressure-Sensitive Multi-Touch Interaction on a Wristband”. In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’15. event-place: Seoul, Republic of Korea. New York, NY, USA: ACM, 2015, pp. 251–254. ISBN: 978-1-4503-3146-3. DOI: 10.1145/2702613.2725441. URL: http://doi.acm.org/10.1145/2702613.2725441 (visited on 11/14/2019).

[2] Maria Giovanna Albanese et al. “A Technique to Improve Text Editing on Smartphones”. English. In: IEEE Computer Society, Sept. 2022, pp. 1–3. ISBN: 978-1-66544-214-5. DOI: 10.1109/VL/HCC53370.2022.9833120. URL: https://www.computer.org/csdl/proceedings-article/vl-hcc/2022/09833120/1FUSHflcy2I (visited on 08/27/2022).

[3] Ohoud Alharbi et al. “WiseType: A Tablet Keyboard with Color-Coded Visualization and Various Editing Options for Error Correction”. In: *Proceedings of Graphics Interface 2019*. GI 2019. event-place: Kingston, Ontario. Toronto, ON, Canada: Canadian Information Processing Society, 2019, Article 4, 1–10. ISBN: 978-0-9947868-4-5. DOI: 10.20380/GI2019.04.

[4] Jessalyn Alvina et al. “CommandBoard: Creating a General-Purpose Command Gesture Input Space for Soft Keyboard”. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST ’17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 17–28. ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126639. URL: https://doi.org/10.1145/3126594.3126639 (visited on 09/08/2021).

[5] Yomna Aly et al. “Spin-lock Gesture Authentication for Mobile Devices”. In: *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct, MobileHCI 2016*. 2016, pp. 775–782. ISBN: 9781450344135. DOI: 10.1145/2957265.2961863. URL: http://dx.doi.org/10.1145/2957265.2961863.

[6] Toshiyuki Ando et al. “One-Handed Rapid Text Selection and Command Execution Method for Smartphones”. In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI EA ’19. event-place: Glasgow, Scotland Uk. New York, NY, USA: ACM, 2019, LBW0224:1–LBW0224:6. ISBN: 978-1-4503-5971-9. DOI: 10.1145/3290607.3312850. URL: http://doi.acm.org/10.1145/3290607.3312850 (visited on 12/16/2019).

[7] Toshiyuki Ando et al. “Press & Tilt: One-Handed Text Selection and Command Execution on Smartphone”. In: *Proceedings of the 30th Australian Conference on Computer-Human Interaction*. OzCHI ’18. event-place: Melbourne, Australia. New York, NY, USA: ACM, 2018, pp. 401–405. ISBN: 978-1-4503-6188-0. DOI: 10.1145/3292147.3292178. URL: http://doi.acm.org/10.1145/3292147.3292178 (visited on 12/16/2019).

[8] Ahmed Sabbir Arif. “A Survey on Mobile Text Entry Handedness: How Do Users Input Text on Handheld Devices While Nomadic?” In: *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*. Dec. 2012, pp. 1–6. DOI: 10.1109/IHCI.2012.6481818.

[9] Ahmed Sabbir Arif. “Metrics for Multi-Touch Input Technologies”. In: *arXiv:2009.13219 [cs]* 2020-09-28 (2010). arXiv: 2009.13219. URL: http://arxiv.org/abs/2009.13219 (visited on 02/27/2021).

[10] Ahmed Sabbir Arif. "Predicting and Reducing the Impact of Errors in Character-Based Text Entry". In: (May 29, 2013). URL: http://hdl.handle.net/10315/28170 (visited on 04/06/2024).

[11] Ahmed Sabbir Arif. "Statistical Grounding". In: *Intelligent Computing for Interactive System Design: Statistics, Digital Signal Processing, and Machine Learning in Practice*. 1st ed. New York, NY, USA: Association for Computing Machinery, 2021, pp. 59–99. ISBN: 978-1-4503-9029-3. URL: https://doi.org/10.1145/3447404.3447410.

[12] Ahmed Sabbir Arif, Benedikt Iltisberger, and Wolfgang Stuerzlinger. "Extending Mobile User Ambient Awareness for Nomadic Text Entry". In: *Proceedings of the 23rd Australian Computer-Human Interaction Conference*. OzCHI '11. New York, NY, USA: ACM, 2011, pp. 21–30. ISBN: 978-1-4503-1090-1. DOI: 10.1145/2071536.2071539. URL: http://doi.acm.org/10.1145/2071536.2071539 (visited on 11/11/2019).

[13] Ahmed Sabbir Arif, Mauricio Herrera López, and Wolfgang Stuerzlinger. "Two New Mobile Touchscreen Text Entry Techniques". In: *Poster at the 36th Graphics Interface Conference*. GI '10. Toronto, ON, Canada: CEUR-WS.org/Vol-588, 2010, pp. 22–23.

[14] Ahmed Sabbir Arif and Ali Mazalek. "A Survey of Text Entry Techniques for Smartwatches". en. In: *Human-Computer Interaction. Interaction Platforms and Techniques*. Ed. by Masaaki Kurosu. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 255–267. ISBN: 978-3-319-39516-6. DOI: 10.1007/978-3-319-39516-6_24.

[15] Ahmed Sabbir Arif and Ali Mazalek. "A Survey of Text Entry Techniques for Smartwatches". en. In: *Human-Computer Interaction. Interaction Platforms and Techniques*. Ed. by Masaaki Kurosu. Vol. 9732. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 255–267. ISBN: 978-3-319-39515-9 978-3-319-39516-6. DOI: 10.1007/978-3-319-39516-6_24. URL: http://link.springer.com/10.1007/978-3-319-39516-6_24 (visited on 08/08/2021).

[16] Ahmed Sabbir Arif, Ali Mazalek, and Wolfgang Stuerzlinger. "The Use of Pseudo Pressure in Authenticating Smartphone Users". In: *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. MOBIQUITOUS '14. Brussels, Belgium: ICST, Dec. 2014, pp. 151–160. ISBN: 978-1-63190-039-6. DOI: 10.4108/icst.mobiquitous.2014.257919. URL: http://dl.acm.org/citation.cfm?id=2692983.2693003 (visited on 10/08/2019).

[17] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. "Analysis of Text Entry Performance Metrics". In: *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*. Washington, DC, USA: IEEE, Sept. 2009, pp. 100–105. DOI: 10.1109/TIC-STH.2009.5444533.

[18] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. "Evaluation of a New Error Prevention Technique for Mobile Touchscreen Text Entry". In: *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*. OzCHI '13. New York, NY, USA: Association for Computing Machinery, Nov. 2013, pp. 397–400. ISBN: 978-1-4503-2525-7. DOI: 10.1145/2541016.2541063. URL: https://doi.org/10.1145/2541016.2541063 (visited on 09/07/2021).

[19] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. "Pseudo-Pressure Detection and Its Use in Predictive Text Entry on Touchscreens". In: *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*. OzCHI '13. New York, NY, USA: ACM, 2013, pp. 383–392. ISBN: 978-1-4503-2525-7. DOI: 10.1145/2541016.2541024. URL: http://doi.acm.org/10.1145/2541016.2541024 (visited on 10/14/2019).

[20] Ahmed Sabbir Arif, Cristina Sylla, and Ali Mazalek. "Learning New Words and Spelling with Autocorrections". In: *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*. ISS '16. event-place: Niagara Falls, Ontario, Canada. New York, NY, USA: ACM, 2016, pp. 409–414. ISBN: 978-1-4503-4248-3. DOI: 10.1145/2992154.2996790. URL: http://doi.acm.org/10.1145/2992154.2996790 (visited on 11/11/2019).

[21] Ahmed Sabbir Arif et al. “Evaluation of a Smart-Restorable Backspace Technique to Facilitate Text Entry Error Correction”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. event-place: San Jose, California, USA. New York, NY, USA: ACM, 2016, pp. 5151–5162. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858407. URL: http://doi.acm.org/10.1145/2858036.2858407 (visited on 11/11/2019).

[22] Ahmed Sabbir Arif et al. “Sparse Tangibles: Collaborative Exploration of Gene Networks Using Active Tangibles and Interactive Tabletops”. In: *Proceedings of the TEI ’16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI ’16. event-place: Eindhoven, Netherlands. New York, NY, USA: ACM, 2016, pp. 287–295. ISBN: 978-1-4503-3582-9. DOI: 10.1145/2839462.2839500. URL: http://doi.acm.org/10.1145/2839462.2839500 (visited on 11/11/2019).

[23] Tyler Baldwin and Joyce Chai. “Towards Online Adaptation and Personalization of Key-Target Resizing for Mobile Devices”. In: *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. IUI ’12. New York, NY, USA: Association for Computing Machinery, Feb. 2012, pp. 11–20. ISBN: 978-1-4503-1048-2. DOI: 10.1145/2166966.2166969. URL: https://dl.acm.org/doi/10.1145/2166966.2166969 (visited on 05/06/2023).

[24] Leon Barnard et al. “An Empirical Comparison of Use-in-Motion Evaluation Scenarios for Mobile Computing Devices”. en. In: *International Journal of Human-Computer Studies* 62.4 (Apr. 2005), pp. 487–520. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2004.12.002. URL: http://www.sciencedirect.com/science/article/pii/S1071581905000145 (visited on 05/06/2020).

[25] Alex Bellos. *The World’s Most Popular Numbers [Excerpt]*. en. June 2014. URL: https://www.scientificamerican.com/article/most-popular-numbers-grapes-of-math/ (visited on 04/20/2021).

[26] Joanna Bergstrom-Lehtovirta, Antti Oulasvirta, and Stephen Brewster. “The Effects of Walking Speed on Target Acquisition on a Touchscreen Interface”. In: *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. MobileHCI ’11. Stockholm, Sweden: Association for Computing Machinery, Aug. 2011, pp. 143–146. ISBN: 978-1-4503-0541-9. DOI: 10.1145/2037373.2037396. URL: https://doi.org/10.1145/2037373.2037396 (visited on 05/05/2020).

[27] Alan F. Blackwell. “The Reification of Metaphor as a Design Tool”. In: *ACM Transactions on Computer-Human Interaction* 13.4 (Dec. 2006), pp. 490–530. ISSN: 1073-0516. DOI: 10.1145/1188816.1188820. URL: https://doi.org/10.1145/1188816.1188820 (visited on 12/14/2022).

[28] Vladimir V. Bochkarev, Anna V. Shevlyakova, and Valery D. Solovyev. *Average Word Length Dynamics as Indicator of Cultural Changes in Society*. arXiv:1208.6109 [cs]. Aug. 2012. DOI: 10.48550/arXiv.1208.6109. URL: http://arxiv.org/abs/1208.6109 (visited on 12/03/2022).

[29] Dieter Bohn. *Here’s what 3D Touch can do on the iPhone 6S*. en. Sept. 2015. URL: https://www.theverge.com/2015/9/22/9370739/3d-touch-features-iphone-6s (visited on 11/11/2019).

[30] Stephen A. Brewster and Michael Hughes. “Pressure-Based Text Entry for Mobile Devices”. In: *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI ’09. event-place: Bonn, Germany. New York, NY, USA: ACM, 2009, 9:1–9:4. ISBN: 978-1-60558-281-8. DOI: 10.1145/1613858.1613870. URL: http://doi.acm.org/10.1145/1613858.1613870 (visited on 11/14/2019).

[31] John Brooke. “SUS - a Quick and Dirty Usability Scale”. In: *Usability Evaluation in Industry* 189.194 (1996), pp. 4–7. DOI: 10.1371/journal.pone.0170531.

[32] Francesco Cafaro, Leilah Lyons, and Alissa N. Antle. “Framed Guessability: Improving the Discoverability of Gestures and Body Movements for Full-Body Interaction”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 1–12. ISBN: 978-1-4503-5620-6. URL: https://doi.org/10.1145/3173574.3174167 (visited on 08/26/2021).

[33] Caroline Cakebread. *Here's How People Are Using Their Smartwatches*. Aug. 2017. URL: https://www.businessinsider.com/most-used-smartwatch-features-chart-2017-8 (visited on 09/17/2020).

[34] Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. USA: L. Erlbaum Associates Inc., 2000. ISBN: 978-0-89859-859-9.

[35] Géry Casiez et al. "The Impact of Control-Display Gain on User Performance in Pointing Tasks". In: *Human–Computer Interaction* 23.3 (Aug. 2008). Publisher: Taylor & Francis, pp. 215–250. ISSN: 0737-0024. DOI: 10.1080/07370020802278163. URL: https://www.tandfonline.com/doi/abs/10.1080/07370020802278163 (visited on 02/05/2021).

[36] Steven J. Castellucci et al. "TiltWriter: Design and Evaluation of a No-Touch Tilt-Based Text Entry Method for Handheld Devices". In: *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*. MUM '19. Article 7. ACM, Nov. 2019, pp. 1–8. ISBN: 978-1-4503-7624-2. DOI: 10.1145/3365610.3365629. URL: http://dl.acm.org/citation.cfm?id=3365610.3365629 (visited on 12/25/2019).

[37] Laura Ceci. *Top Smartphone Users Activities 2022*. en. Jan. 2024. URL: https://www.statista.com/statistics/1337895/top-smartphone-activities/ (visited on 04/11/2024).

[38] A. Chamberlain and R. Kalawsky. "A Comparative Investigation into Two Pointing Systems for Use with Wearable Computers While Mobile". In: *Eighth International Symposium on Wearable Computers*. Vol. 1. ISSN: 1530-0811. Oct. 2004, pp. 110–117. DOI: 10.1109/ISWC.2004.1.

[39] Youli Chang et al. "Understanding Users' Touch Behavior on Large Mobile Touch-Screens and Assisted Targeting by Tilting Gesture". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 1499–1508. ISBN: 9781450331456. URL: https://doi.org/10.1145/2702123.2702425.

[40] Chen Chen et al. "BezelCopy: An Efficient Cross-Application Copy-Paste Technique for Touchscreen Smartphones". In: *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*. AVI '14. New York, NY, USA: Association for Computing Machinery, May 2014, pp. 185–192. ISBN: 978-1-4503-2775-6. DOI: 10.1145/2598153.2598162. URL: https://doi.org/10.1145/2598153.2598162 (visited on 09/04/2021).

[41] Xiang 'Anthony' Chen, Tovi Grossman, and George Fitzmaurice. "Swipeboard: A Text Entry Technique for Ultra-Small Interfaces That Supports Novice to Expert Transitions". In: *Proceedings of the 27th Annual Acm Symposium on User Interface Software and Technology*. UIST '14. ACM, Oct. 2014, pp. 615–620. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647354. URL: http://dl.acm.org/citation.cfm?id=2642918.2647354 (visited on 11/22/2019).

[42] Xiang 'Anthony' Chen et al. "Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. New York, NY, USA: Association for Computing Machinery, Apr. 2014, pp. 159–168. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2556955. URL: https://doi.org/10.1145/2556288.2556955 (visited on 04/20/2021).

[43] Jason Cipriani. *Google's Secret Keyboard Feature Lets You Easily Move the Cursor*. en. May 2016. URL: https://www.cnet.com/tech/services-and-software/googles-secret-keyboard-feature-gives-you-precise-cursor-control/ (visited on 09/06/2021).

[44] A. Cockburn, D. Ahlström, and C. Gutwin. "Understanding Performance in Touch Selections: Tap, Drag and Radial Pointing Drag with Finger, Stylus and Mouse". en. In: *International Journal of Human-Computer Studies* 70.3 (Mar. 2012), pp. 218–233. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2011.11.002. URL: https://www.sciencedirect.com/science/article/pii/S1071581911001546 (visited on 02/05/2021).

[45] J Cohen. *Statistical Power Analysis for the Behavioral Sciences*. 2nd. Hillsdale, NJ, USA: Lawrence Erlbaum, 1988, p. 283. ISBN: 0805802835. DOI: 10.1234/12345678.

[46] Christian Corsten et al. "ForceRay: Extending Thumb Reach via Force Input Stabilizes Device Grip for Mobile Touch Input". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19* Chi (2019), pp. 1–12. DOI: 10.1145/3290605.3300442. URL: http://dl.acm.org/citation.cfm?doid=3290605.3300442.

[47] Christian Corsten et al. "Use the Force Picker, Luke: Space-Efficient Value Input on Force-Sensitive Mobile Touchscreens". In: *Conference on Human Factors in Computing Systems - Proceedings*. Vol. 2018-April. 2018. ISBN: 9781450356206. DOI: 10.1145/3173574.3174235. URL: https://doi.org/10.1145/3173574.3174235.

[48] Rajkumar Darbar, Prasanta Kr Sen, and Debasis Samanta. "Presstact: Side Pressure-Based Input for Smartwatch Interaction". In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA '16. event-place: San Jose, California, USA. New York, NY, USA: ACM, 2016, pp. 2431–2438. ISBN: 978-1-4503-4082-3. DOI: 10.1145/2851581.2892436. URL: http://doi.acm.org/10.1145/2851581.2892436 (visited on 11/14/2019).

[49] Rajkumar Darbar et al. "Exploring Smartphone-Enabled Text Selection in AR-HMD". In: *Proceedings of Graphics Interface 2021*. GI 2021. ISSN: 0713-5424 event-place: Virtual Event. Toronto, ON, Canada: Canadian Information Processing Society, 2021, pp. 117–126. ISBN: 978-0-9947868-6-9. DOI: 10.20380/GI2021.14.

[50] Sean DeLong, Ahmed Sabbir Arif, and Ali Mazalek. "Design and Evaluation of Graphical Feedback on Tangible Interactions in a Low-Resolution Edge Display". In: ACM, June 2019, p. 8. ISBN: 978-1-4503-6751-6. DOI: 10.1145/3321335.3324954. URL: http://dl.acm.org/citation.cfm?id=3321335.3324954 (visited on 10/08/2019).

[51] Statista Research Department. *Global: Smartwatches Market Revenue 2019-2028*. en. 2024. URL: https://www.statista.com/forecasts/1314322/worldwide-revenue-of-smartwatch-market (visited on 04/11/2024).

[52] Michael Dickens. *Which Numbers are the Most Common?* en. Feb. 2011. URL: https://mathematicalmulticore.wordpress.com/2011/02/04/which-numbers-are-the-most-common/ (visited on 04/20/2021).

[53] Luke Dormehl. *iPhone UI Designer Tells The Story Behind iOS Text Selection Patent*. en-US. Section: News. Feb. 2014. URL: https://www.cultofmac.com/266587/iphone-ui-designer-tells-story-behind-ios-text-selection-patent/ (visited on 04/11/2024).

[54] Mark D. Dunlop, Marc Roper, and Gennaro Imperatore. "Text Entry Tap Accuracy and Exploration of Tilt Controlled Layered Interaction on Smartwatches". In: *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI '17. New York, NY, USA: Association for Computing Machinery, Sept. 2017, pp. 1–11. ISBN: 978-1-4503-5075-4. DOI: 10.1145/3098279.3098560. URL: https://doi.org/10.1145/3098279.3098560 (visited on 09/13/2020).

[55] Lisa Eadicicco. *New Details About The Screen For Apple's Next iPhone May Have Just Leaked*. Jan. 2015. URL: https://www.businessinsider.com/iphone-6s-rumors-screen-2015-1 (visited on 05/05/2020).

[56] Alexander Keith Eady and Audrey Girouard. "Caret Manipulation using Deformable Input in Mobile Devices". In: *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI '15. New York, NY, USA: Association for Computing Machinery, Jan. 2015, pp. 587–591. ISBN: 978-1-4503-3305-4. DOI: 10.1145/2677199.2687916. URL: https://doi.org/10.1145/2677199.2687916 (visited on 09/08/2021).

[57] Brien East et al. "Actibles: Open Source Active Tangibles". en. In: *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces - ISS '16*. Niagara Falls, Ontario, Canada: ACM Press, 2016, pp. 469–472. ISBN: 978-1-4503-4248-3. DOI: 10.1145/2992154.2996874. URL: http://dl.acm.org/citation.cfm?doid=2992154.2996874 (visited on 08/03/2020).

[58] Katherine Fennedy et al. “Towards a Unified and Efficient Command Selection Mechanism for Touch-Based Devices Using Soft Keyboard Hotkeys”. In: *ACM Transactions on Computer-Human Interaction* (2021). Publisher: Association for Computing Machinery. URL: https://hal.archives-ouvertes.fr/hal-03319260 (visited on 08/24/2021).

[59] Angus Graeme Forbes and Tony Fast. “The Natural Materials Browser: Using a Tablet Interface for Exploring Volumetric Materials Science Datasets”. en. In: (), p. 2.

[60] Vittorio Fuccella, Poika Isokoski, and Benoit Martin. “Gestures and Widgets: Performance in Text Editing on Multi-Touch Capable Mobile Devices”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. New York, NY, USA: Association for Computing Machinery, Apr. 2013, pp. 2785–2794. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481385. URL: https://doi.org/10.1145/2470654.2481385 (visited on 05/16/2021).

[61] Vittorio Fuccella and Benoît Martin. “TouchTap: A Gestural Technique to Edit Text on Multi-Touch Capable Mobile Devices”. In: *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter*. CHItaly ’17. New York, NY, USA: Association for Computing Machinery, Sept. 2017, pp. 1–6. ISBN: 978-1-4503-5237-6. DOI: 10.1145/3125571.3125579. URL: https://doi.org/10.1145/3125571.3125579 (visited on 05/16/2021).

[62] Jun Gong, Bryan Haggerty, and Peter Tarasewich. “An Enhanced Multitap Text Entry Method with Predictive Next-Letter Highlighting”. In: *CHI ’05 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’05. New York, NY, USA: Association for Computing Machinery, Apr. 2005, pp. 1399–1402. ISBN: 978-1-59593-002-6. DOI: 10.1145/1056808.1056926. URL: https://dl.acm.org/doi/10.1145/1056808.1056926 (visited on 05/06/2023).

[63] Jun Gong et al. “WrisText: One-Handed Text Entry on Smartwatch Using Wrist Gestures”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. event-place: Montreal QC, Canada. New York, NY, USA: ACM, 2018, 181:1–181:14. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173755. URL: http://doi.acm.org/10.1145/3173574.3173755 (visited on 12/26/2019).

[64] Mitchell Gordon, Tom Ouyang, and Shumin Zhai. “WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. event-place: San Jose, California, USA. New York, NY, USA: ACM, 2016, pp. 3817–3821. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858242. URL: http://doi.acm.org/10.1145/2858036.2858242 (visited on 12/26/2019).

[65] Jens Grubert et al. “MultiFi: Multi Fidelity Interaction with Displays On and Around the Body”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. New York, NY, USA: Association for Computing Machinery, Apr. 2015, pp. 3933–3942. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702331. URL: https://doi.org/10.1145/2702123.2702331 (visited on 09/28/2020).

[66] Asela Gunawardana, Tim Paek, and Christopher Meek. “Usability Guided Key-Target Resizing for Soft Keyboards”. In: *Proceedings of the 15th international conference on Intelligent user interfaces*. IUI ’10. New York, NY, USA: Association for Computing Machinery, Feb. 2010, pp. 111–118. ISBN: 978-1-60558-515-4. DOI: 10.1145/1719970.1719986. URL: https://doi.org/10.1145/1719970.1719986 (visited on 01/28/2023).

[67] Aakar Gupta and Ravin Balakrishnan. “DualKey: Miniature Screen Text Entry via Finger Identification”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. San Jose, California, USA: Association for Computing Machinery, May 2016, pp. 59–70. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858052. URL: https://doi.org/10.1145/2858036.2858052 (visited on 05/03/2020).

[68] Carl Gutwin et al. “Learning Multiple Mappings: an Evaluation of Interference, Transfer, and Retention with Chorded Shortcut Buttons”. In: *Proceedings of Graphics Interface 2020*. GI 2020. event-place: University of Toronto. Toronto, Ontario, Canada: Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 2020, pp. 206–214. ISBN: 978-0-9947868-5-2. DOI: 10.20380/GI2020.21.

[69] Kiyotaka Hara, Takeshi Umezawa, and Noritaka Osawa. “Effect of Button Size and Location When Pointing with Index Finger on Smartwatch”. en. In: *Human-Computer Interaction: Interaction Technologies*. Ed. by Masaaki Kurosu. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 165–174. ISBN: 978-3-319-20916-6. DOI: 10.1007/978-3-319-20916-6_16.

[70] Aurora Harley. *Touch Targets on Touchscreens*. en. May 2019. URL: https://www.nngroup.com/articles/touch-target-size/ (visited on 08/02/2022).

[71] Sandra G. Hart. “NASA-task Load Index (NASA-TLX); 20 Years Later”. In: *Proceedings of the human factors and ergonomics society annual meeting*. Vol. 50. 9. Sage publications Sage CA: Los Angeles, CA. 2006, pp. 904–908.

[72] Sandra G. Hart. “Nasa-Task Load Index (NASA-TLX); 20 Years Later”. en. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50.9 (Oct. 2006). Publisher: SAGE Publications Inc, pp. 904–908. ISSN: 2169-5067. DOI: 10.1177/154193120605000909. URL: https://doi.org/10.1177/154193120605000909 (visited on 11/22/2020).

[73] Sandra G. Hart and Lowell E. Staveland. “Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research”. en. In: *Advances in Psychology*. Vol. 52. Amsterdam, The Netherlands: Elsevier, 1988, pp. 139–183. ISBN: 978-0-444-70388-0. DOI: 10.1016/S0166-4115(08)62386-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166411508623869 (visited on 11/22/2020).

[74] Android Accessibility Help. *Touch Target Size - Android Accessibility Help*. 2022. URL: https://support.google.com/accessibility/android/answer/7101858?hl=en (visited on 08/02/2022).

[75] Uta Hinrichs and Sheelagh Carpendale. “Gestures in The Wild: Studying Multi-Touch Gesture Sequences on Interactive Tabletop Exhibits”. In: *Conference on Human Factors in Computing Systems - Proceedings*. 2011, pp. 3023–3032. ISBN: 9781450302289. DOI: 10.1145/1978942.1979391. URL: http://www.ideum.com/products/multitouch/.

[76] Jonggi Hong et al. “SplitBoard: A Simple Split Soft Keyboard for Wristwatch-Sized Touch Screens”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. event-place: Seoul, Republic of Korea. New York, NY, USA: ACM, 2015, pp. 1233–1236. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702273. URL: http://doi.acm.org/10.1145/2702123.2702273 (visited on 11/22/2019).

[77] Apple Inc. *Buttons - Menus and actions - Components - Human Interface Guidelines - Design - Apple Developer*. 2022. URL: https://developer.apple.com/design/human-interface-guidelines/components/menus-and-actions/buttons/ (visited on 08/02/2022).

[78] International Organization for Standardization. *ISO/TS 9241-411:2012*. en. May 2012. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/41/54106.html (visited on 08/21/2021).

[79] Akira Ishii and Buntarou Shizuki. “Exploring Callout Design in Selection Task for Ultra-Small Touch Screen Devices”. In: *Proceedings of the 28th Australian Conference on Computer-Human Interaction*. OzCHI ’16. New York, NY, USA: Association for Computing Machinery, Nov. 2016, pp. 426–434. ISBN: 978-1-4503-4618-4. DOI: 10.1145/3010915.3010922. URL: https://doi.org/10.1145/3010915.3010922 (visited on 02/13/2022).

[80] Christina L. James and Kelly M. Reischel. “Text Input for Mobile Devices: Comparing Model Prediction to Actual Performance”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’01. New York, NY, USA: Association for Computing Machinery, Mar. 2001, pp. 365–371. ISBN: 978-1-58113-327-1. DOI: 10.1145/365024.365300. URL: https://doi.org/10.1145/365024.365300 (visited on 10/25/2022).

[81] T. Kaaresoja and J. Linjama. “Perception of Short Tactile Pulses Generated by a Vibration Motor in a Mobile Phone”. In: *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics Conference*. Mar. 2005, pp. 471–472. DOI: 10.1109/WHC.2005.103.

[82] Tomonari Kamba et al. “Using Small Screen Space More Efficiently”. en. In: *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI ’96*. Vancouver, British Columbia, Canada: ACM Press, 1996, pp. 383–390. ISBN: 978-0-89791-777-3. DOI: 10.1145/238386.238582. URL: http://portal.acm.org/citation.cfm?doid=238386.238582 (visited on 11/19/2019).

[83] Erik D. Kennedy. *The Android/Material Design Font Size Guidelines*. Apr. 2018. URL: https://learnui.design/blog/android-material-design-font-size-guidelines.html (visited on 09/08/2021).

[84] Jungsoo Kim et al. “The Gesture Watch: A Wireless Contact-free Gesture based Wrist Interface”. In: *2007 11th IEEE International Symposium on Wearable Computers*. IEEE. 2007, pp. 15–22.

[85] Andreas Komninos and Mark Dunlop. “Text Input on a Smart Watch”. In: *IEEE Pervasive Computing* 13.4 (Oct. 2014). Conference Name: IEEE Pervasive Computing, pp. 50–58. ISSN: 1558-2590. DOI: 10.1109/MPRV.2014.77.

[86] Andreas Komninos et al. “A Glimpse of Mobile Text Entry Errors and Corrective Behaviour in the Wild”. In: *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. MobileHCI ’18. Barcelona, Spain: Association for Computing Machinery, Sept. 2018, pp. 221–228. ISBN: 978-1-4503-5941-2. DOI: 10.1145/3236112.3236143. URL: https://doi.org/10.1145/3236112.3236143 (visited on 05/17/2020).

[87] Sven Kratz and Michael Rohs. “HoverFlow: Exploring Around-Device Interaction with Ir Distance Sensors”. In: *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI ’09. New York, NY, USA: Association for Computing Machinery, Sept. 2009, pp. 1–4. ISBN: 978-1-60558-281-8. DOI: 10.1145/1613858.1613912. URL: https://doi.org/10.1145/1613858.1613912 (visited on 09/17/2020).

[88] Satvik Kulshreshtha and Ahmed Sabbir Arif. “Text Entry Performance on an Expandable Socket Attached Smartphone in Stationary and Mobile Settings”. en. In: *Advances in Usability and User Experience*. Ed. by Tareq Ahram and Christianne Falcão. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2020, pp. 207–217. ISBN: 978-3-030-19135-1. DOI: 10.1007/978-3-030-19135-1_21.

[89] Hiroki Kurosawa, Daisuke Sakamoto, and Tetsuo Ono. “MyoTilt: A Target Selection Method for Smartwatches Using the Tilting Operation and Electromyography”. In: *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI ’18. New York, NY, USA: Association for Computing Machinery, Sept. 2018, pp. 1–11. ISBN: 978-1-4503-5898-9. DOI: 10.1145/3229434.3229457. URL: https://doi.org/10.1145/3229434.3229457 (visited on 08/03/2022).

[90] Paul Lamkin. *Smart Wearables Market to Double by 2022: $27 Billion Industry Forecast*. International Data Corporation (IDC). Link to the article. Oct. 2018. URL: https://www.forbes.com/sites/paullamkin/2018/10/23/smart-wearables-market-to-double-by-2022-27-billion-industry-forecast/?sh=2a1c63442656 (visited on 12/17/2020).

[91] Federica Laricchia. *Smartphone Market Shares by Vendor 2009-2023*. en. Feb. 2024. URL: https://www.statista.com/statistics/271496/global-market-share-held-by-smartphone-vendors-since-4th-quarter-2009/ (visited on 04/11/2024).

[92] Huy Viet Le et al. "Shortcut Gestures for Mobile Text Editing on Fully Touch Sensitive Smartphones". In: *ACM Transactions on Computer-Human Interaction* 27.5 (Aug. 2020), 33:1–33:38. ISSN: 1073-0516. DOI: 10.1145/3396233. URL: https://doi.org/10.1145/3396233 (visited on 08/16/2021).

[93] Reena Lee. *Gboard, Now Available for Android*. en. Library Catalog: blog.google. Dec. 2016. URL: https://blog.google/products/search/gboard-now-on-android/ (visited on 03/02/2020).

[94] Luis A. Leiva et al. "Text Entry on Tiny QWERTY Soft Keyboards". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, Apr. 2015, pp. 669–678. ISBN: 978-1-4503-3145-6. URL: https://doi.org/10.1145/2702123.2702388 (visited on 02/04/2021).

[95] Vladimir I Levenshtein et al. "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals". In: *Soviet physics doklady*. Vol. 10. Issue: 8. Soviet Union: MAIK Nauka/Interperiodica, 1966, pp. 707–710.

[96] James R. Lewis. "The System Usability Scale: Past, Present, and Future". en. In: *International Journal of Human–Computer Interaction* 34.7 (July 2018), pp. 577–590. ISSN: 1044-7318, 1532-7590. DOI: 10.1080/10447318.2018.1455307. URL: https://www.tandfonline.com/doi/full/10.1080/10447318.2018.1455307 (visited on 07/06/2021).

[97] O J Lewis, R J Hamshere, and T M Bucknill. "The Anatomy of the Wrist Joint". In: *Journal of Anatomy* 106.Pt 3 (May 1970), pp. 539–552. ISSN: 0021-8782. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1233428/ (visited on 09/14/2020).

[98] Christian Loclair, Sean Gustafson, and Patrick Baudisch. "PinchWatch: A Wearable Device for One-Handed Microinteractions". In: *Proceedings of the 12th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI '10. Association for Computing Machinery, 2010.

[99] J. C. Lupis. *These Are the Smartwatch Functions People Use the Most*. Marketing Charts. Aug. 21, 2017. URL: https://www.marketingcharts.com/digital/non-mobile-connected-devices-79748 (visited on 04/06/2024).

[100] I. Scott MacKenzie. "Fitts' Law". en. In: *The Wiley Handbook of Human Computer Interaction*. Hoboken, NJ, USA: John Wiley & Sons, Ltd, 2018, pp. 347–370. ISBN: 978-1-118-97600-5. DOI: 10.1002/9781118976005.ch17. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118976005.ch17 (visited on 06/11/2021).

[101] I. Scott MacKenzie. *Human-Computer Interaction: An Empirical Research Perspective*. en. First edition. Amsterdam: Morgan Kaufmann is an imprint of Elsevier, 2013. ISBN: 978-0-12-405865-1.

[102] David C. McCallum et al. "PressureText: Pressure Input for Mobile Phone Text Entry". In: *CHI '09 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '09. New York, NY, USA: Association for Computing Machinery, Apr. 2009, pp. 4519–4524. ISBN: 978-1-60558-247-4. DOI: 10.1145/1520340.1520693. URL: https://doi.org/10.1145/1520340.1520693 (visited on 08/03/2022).

[103] Meghna Mehta et al. "Active Pathways: Using Active Tangibles and Interactive Tabletops for Collaborative Modeling in Systems Biology". In: *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*. ISS '16. event-place: Niagara Falls, Ontario, Canada. New York, NY, USA: ACM, 2016, pp. 129–138. ISBN: 978-1-4503-4248-3. DOI: 10.1145/2992154.2992176. URL: http://doi.acm.org/10.1145/2992154.2992176 (visited on 11/11/2019).

[104] Mudit Misra, Elena Márquez Segura, and Ahmed Sabbir Arif. "Exploring the Pace of an Endless Runner Game in Stationary and Mobile Settings". In: *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*. CHI PLAY '19 Extended Abstracts. Barcelona, Spain: Association for Computing Machinery, Oct. 2019, pp. 543–550. ISBN: 978-1-4503-6871-1. DOI: 10.1145/3341215.3356256. URL: https://doi.org/10.1145/3341215.3356256 (visited on 02/16/2020).

[105] Sachi Mizobuchi, Mark Chignell, and David Newton. "Mobile Text Entry: Relationship Between Walking Speed and Text Input Task Difficulty". In: *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*. MobileHCI '05. Salzburg, Austria: Association for Computing Machinery, Sept. 2005, pp. 122–128. ISBN: 978-1-59593-089-7. DOI: 10.1145/1085777.1085798. URL: https://doi.org/10.1145/1085777.1085798 (visited on 05/05/2020).

[106] Andy Moser. *The Simple iPhone Hack You Need to Fix Your Texting Typos*. en. Section: Tech. Feb. 2021. URL: https://mashable.com/article/how-to-use-iphone-keyboard-trackpad-texting (visited on 12/14/2022).

[107] NASA. *NASA Task Load Index (TLX) V 1.0: Paper and Pencil Package*. Tech. rep. Moffett Field, CA, USA: Human Performance Research Group, NASA Ames Research Center, 1986.

[108] Yusuke Niiro, Marcelo Kallmann, and Ahmed Sabbir Arif. "An Experimental Comparison of Touch and Pen Gestures on a Vertical Display". In: *Proceedings of the 8th ACM International Symposium on Pervasive Displays*. PerDis '19. New York, NY, USA: Association for Computing Machinery, June 2019, pp. 1–6. ISBN: 978-1-4503-6751-6. DOI: 10.1145/3321335.3324936. URL: https://doi.org/10.1145/3321335.3324936 (visited on 08/14/2020).

[109] Peter Norvig. *English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU*. 2013. URL: http://norvig.com/mayzner.html (visited on 12/14/2022).

[110] Stephen Oney et al. "ZoomBoard: A Diminutive Qwerty Soft Keyboard Using Iterative Zooming for Ultra-Small Devices". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. event-place: Paris, France. New York, NY, USA: ACM, 2013, pp. 2799–2802. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481387. URL: http://doi.acm.org/10.1145/2470654.2481387 (visited on 12/26/2019).

[111] Laxmi Pandey, Azar Alizadeh, and Ahmed Sabbir Arif. "Enabling Predictive Number Entry and Editing on Touchscreen-Based Mobile Devices". In: *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*. CHIIR '20. New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 13–22. ISBN: 978-1-4503-6892-6. DOI: 10.1145/3343413.3377957. URL: https://doi.org/10.1145/3343413.3377957 (visited on 09/02/2020).

[112] Benjamin Poppinga et al. "Understanding Shortcut Gestures on Mobile Touch Devices". In: *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services*. MobileHCI '14. Toronto, ON, Canada: Association for Computing Machinery, 2014, pp. 173–182. ISBN: 9781450330046. DOI: 10.1145/2628363.2628378. URL: https://doi.org/10.1145/2628363.2628378.

[113] Philip Quinn and Shumin Zhai. "Modeling Gesture-Typing Movements". In: *Human–Computer Interaction* 33.3 (May 2018). Publisher: Taylor & Francis, pp. 234–280. ISSN: 0737-0024. DOI: 10.1080/07370024.2016.1215922. URL: https://doi.org/10.1080/07370024.2016.1215922 (visited on 04/15/2020).

[114] Gulnar Rakhmetulla and Ahmed Sabbir Arif. "SwipeRing: Gesture Typing on Smartwatches Using a Segmented QWERTY Around the Bezel". In: *Proceedings of Graphics Interface 2021*. GI 2021. ISSN: 0713-5424 event-place: Virtual Event. Toronto, ON, Canada: Canadian Information Processing Society, 2021, pp. 166–177. ISBN: 978-0-9947868-6-9. DOI: 10.20380/GI2021.19.

[115] Gulnar Rakhmetulla and Ahmed Sabbir Arif. “SwipeRing: Gesture Typing on Smartwatches Using a Segmented Qwerty Around the Bezel”. en. In: Dec. 2020. URL: https://openreview.net/forum?id=eVyaNs3sm8a (visited on 04/21/2021).

[116] Gulnar Rakhmetulla et al. “Using Action-Level Metrics to Report the Performance of Multi-Step Keyboards”. In: *Proceedings of Graphics Interface 2021*. GI 2021. ISSN: 0713-5424 event-place: Virtual Event. Toronto, ON, Canada: Canadian Information Processing Society, 2021, pp. 127–137. ISBN: 978-0-9947868-6-9. DOI: 10.20380/GI2021.15.

[117] Volker Roth and Thea Turner. “Bezel Swipe: Conflict-Free Scrolling and Multiple Selection on Mobile Touch Screen Devices”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, Apr. 2009, pp. 1523–1526. ISBN: 978-1-60558-246-7. URL: https://doi.org/10.1145/1518701.1518933 (visited on 09/04/2021).

[118] Judy Sanhz. *How to Use Gboard Clipboard in Android*. en-US. July 2020. URL: https://www.technipages.com/how-to-use-gboard-clipboard-in-android (visited on 09/06/2021).

[119] Jean-Baptiste Scheibel et al. “Virtual Stick in Caret Positioning on Touch Screens”. In: *Proceedings of the 25th Conference on l’Interaction Homme-Machine*. IHM ’13. New York, NY, USA: Association for Computing Machinery, Nov. 2013, pp. 107–114. ISBN: 978-1-4503-2407-6. DOI: 10.1145/2534903.2534918. URL: https://doi.org/10.1145/2534903.2534918 (visited on 09/04/2021).

[120] Robin Schweigert et al. “KnuckleTouch: Enabling Knuckle Gestures on Capacitive Touchscreens using Deep Learning”. In: *Proceedings of Mensch und Computer 2019*. MuC’19. New York, NY, USA: Association for Computing Machinery, Sept. 2019, pp. 387–397. ISBN: 978-1-4503-7198-8. DOI: 10.1145/3340764.3340767. URL: https://doi.org/10.1145/3340764.3340767 (visited on 09/08/2021).

[121] R. W. Soukoreff and I. S. MacKenzie. “An Informatic Rationale for the Speed-Accuracy Trade-Off”. In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. ISSN: 1062-922X. Oct. 2009, pp. 2890–2896. DOI: 10.1109/ICSMC.2009.5346580.

[122] R. William Soukoreff and I. Scott MacKenzie. “Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’03. event-place: Ft. Lauderdale, Florida, USA. New York, NY, USA: ACM, 2003, pp. 113–120. ISBN: 978-1-58113-630-2. DOI: 10.1145/642611.642632. URL: http://doi.acm.org/10.1145/642611.642632 (visited on 11/12/2019).

[123] R. William Soukoreff and I. Scott MacKenzie. “Towards a Standard for Pointing Device Evaluation, Perspectives on 27 Years of Fitts’ Law Research in HCI”. en. In: *International Journal of Human-Computer Studies*. Fitts’ Law 50 Years Later: Applications and Contributions from Human-Computer Interaction 61.6 (Dec. 2004), pp. 751–789. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2004.09.001.

[124] Daniel Spelmezan et al. “Side Pressure for Bidirectional Navigation on Small Devices”. In: (2013). DOI: 10.1145/2493190.2493199. URL: http://dx.doi.org/10.1145/2493190.2493199.

[125] HUI Tang, DAVID J. Beebe, and ARTHUR F. Kramer. “A Multilevel Input System with Force-Sensitive Elements”. en. In: *International Journal of Human-Computer Studies* 54.4 (Apr. 2001), pp. 495–507. ISSN: 1071-5819. DOI: 10.1006/ijhc.2000.0451. URL: https://www.sciencedirect.com/science/article/pii/S1071581900904518 (visited on 08/04/2022).

[126] Larry Tesler. “A Personal History of Modeless Text Editing and Cut/Copy-Paste”. en. In: *Interactions* 19.4 (July 2012), pp. 70–75. ISSN: 1072-5520, 1558-3449. DOI: 10.1145/2212877.2212896. URL: https://dl.acm.org/doi/10.1145/2212877.2212896 (visited on 09/08/2021).

[127] Jessica J Tran et al. “Exploring Pinch and Spread Gestures on Mobile Devices”. In: *MobileHCI 2013 - Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2013, pp. 151–160. ISBN: 9781450322737. DOI: 10.1145/2493190.2493221.

[128] Theophanis Tsandilas et al. "Coordination of Tilt and Touch in One- and Two-Handed Use". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, pp. 2001–2004. ISBN: 9781450324731. DOI: 10.1145/2556288.2557088. URL: https://doi.org/10.1145/2556288.2557088.

[129] Colton J. Turner, Barbara S. Chaparro, and Jibo He. "Texting While Walking: Is It Possible with a Smartwatch?" In: *Journal of Usability Studies* 13.2 (Feb. 2018), pp. 94–118. ISSN: 1931-3357.

[130] Keith Vertanen and Per Ola Kristensson. "A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails". In: *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. MobileHCI '11. New York, NY, USA: Association for Computing Machinery, Aug. 2011, pp. 295–298. ISBN: 978-1-4503-0541-9. DOI: 10.1145/2037373.2037418. URL: https://doi.org/10.1145/2037373.2037418 (visited on 08/02/2022).

[131] Keith Vertanen et al. "VelociWatch: Designing and Evaluating a Virtual Keyboard for the Input of Challenging Text". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Paper 591. New York, NY, USA: ACM, May 2019, pp. 1–14. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300821. URL: http://dl.acm.org/citation.cfm?id=3290605.3300821 (visited on 12/20/2019).

[132] Daniel Vogel and Patrick Baudisch. "Shift: A Technique for Operating Pen-Based Interfaces Using Touch". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. event-place: San Jose, California, USA. New York, NY, USA: ACM, 2007, pp. 657–666. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240727. URL: http://doi.acm.org/10.1145/1240624.1240727 (visited on 11/14/2019).

[133] Emily A. Vogels. *About One-in-Five Americans Use a Smart Watch or Fitness Tracker*. en-US. Jan. 2020. URL: https://www.pewresearch.org/fact-tank/2020/01/09/about-one-in-five-americans-use-a-smart-watch-or-fitness-tracker/ (visited on 09/17/2020).

[134] Feng Wang and Xiangshi Ren. "Empirical Evaluation for Finger Input Properties in Multi-Touch Interaction". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. New York, NY, USA: Association for Computing Machinery, Apr. 2009, pp. 1063–1072. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518864. URL: https://doi.org/10.1145/1518701.1518864 (visited on 12/13/2022).

[135] Katrin Wolf, Robert Schleicher, and Michael Rohs. "Ergonomic Characteristics of Gestures for Front- and Back-of-Tablets Interaction with Grasping Hands". In: *MobileHCI 2014 - Proceedings of the 16th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2014, pp. 453–458. ISBN: 9781450327718. DOI: 10.1145/2628363.2634214. URL: http://dx.doi.org/10.1145/2628363.2634214.

[136] Haijun Xia, Tovi Grossman, and George Fitzmaurice. "NanoStylus: Enhancing Input on Ultra-Small Displays with a Finger-Mounted Stylus". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST '15. New York, NY, USA: Association for Computing Machinery, Nov. 2015, pp. 447–456. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807500. URL: https://doi.org/10.1145/2807442.2807500 (visited on 08/03/2022).

[137] Haijun Xia et al. "Iteratively Designing Gesture Vocabularies: A Survey and Analysis of Best Practices in the HCI Literature". In: *ACM Transactions on Computer-Human Interaction* 29.4 (May 2022), 37:1–37:54. ISSN: 1073-0516. DOI: 10.1145/3503537. URL: https://doi.org/10.1145/3503537 (visited on 12/14/2022).

[138] Chao Xu, Parth H. Pathak, and Prasant Mohapatra. “Finger-Writing with Smartwatch: A Case for Finger and Hand Gesture Recognition Using Smartwatch”. In: *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. HotMobile ’15. Santa Fe, New Mexico, USA: Association for Computing Machinery, 2015, pp. 9–14. ISBN: 9781450333917. DOI: 10.1145/2699343.2699350. URL: https://doi.org/10.1145/2699343.2699350.

[139] Hui-Shyong Yeo et al. “WatchMI: Pressure Touch, Twist and Pan Gesture Input on Unmodified Smartwatches”. In: *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI ’16. event-place: Florence, Italy. New York, NY, USA: ACM, 2016, pp. 394–399. ISBN: 978-1-4503-4408-1. DOI: 10.1145/2935334.2935375. URL: http://doi.acm.org/10.1145/2935334.2935375 (visited on 11/14/2019).

[140] Xin Yi et al. “Is It Too Small?: Investigating the Performances and Preferences of Users When Typing on Tiny Qwerty Keyboards”. en. In: *International Journal of Human-Computer Studies* 106 (Oct. 2017), pp. 44–62. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2017.05.001. URL: http://www.sciencedirect.com/science/article/pii/S1071581917300654 (visited on 11/22/2019).

[141] Shumin Zhai and Per Ola Kristensson. “The Word-Gesture Keyboard: Reimagining Keyboard Interaction”. In: *Communications of the ACM* 55.9 (Sept. 2012), pp. 91–101. ISSN: 0001-0782. DOI: 10.1145/2330667.2330689. URL: https://doi.org/10.1145/2330667.2330689 (visited on 05/03/2020).

[142] Mingrui Zhang and Jacob O. Wobbrock. “Gedit: Keyboard Gestures for Mobile Text Editing”. In: *Proceedings of Graphics Interface 2020*. GI 2020. event-place: University of Toronto. Toronto, Ontario, Canada: Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 2020, pp. 470–473. ISBN: 978-0-9947868-5-2. DOI: 10.20380/GI2020.47.

[143] Maozheng Zhao et al. “Voice and Touch Based Error-tolerant Multimodal Text Editing and Correction for Smartphones”. en. In: *Proceedings of the ACM Symposium on User Interface Software and Technology*. UIST ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 17.

[144] Mingyuan Zhong et al. “ForceBoard: Subtle Text Entry Leveraging Pressure”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 1–10. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174102. URL: https://doi.org/10.1145/3173574.3174102 (visited on 10/24/2022).