Shortcut Gestures for Mobile Text Editing on Fully Touch Sensitive Smartphones

HUY VIET LE, University of Stuttgart
SVEN MAYER, Carnegie Mellon University
MAXIMILIAN WEIß, JONAS VOGELSANG, and HENRIKE WEINGÄRTNER, University of Stuttgart
NIELS HENZE, University of Regensburg

While advances in mobile text entry enable smartphone users to type almost as fast as on hardware keyboards, text-heavy activities are still not widely adopted. One reason is the lack of shortcut mechanisms. In this article, we determine shortcuts for text-heavy activities, elicit shortcut gestures, implement them for a fully touch-sensitive smartphone, and conduct an evaluation with potential users. We found that experts perform around 800 keyboard shortcuts per day, which are not available on smartphones. Interviews revealed the lack of shortcuts as a major limitation that prevents mobile text editing. Therefore, we elicited gestures for the 22 most important shortcuts for smartphones that are touch-sensitive on the whole device surface. We implemented the gestures for a fully touch-sensitive smartphone using deep learning and evaluated them in realistic scenarios to gather feedback. We show that the developed prototype is perceived as intuitive and faster than recent commercial approaches.

CCS Concepts: • **Human-centered computing** → **Gestural input**; *Touch screens*; *Empirical studies in HCI*; *Smartphones*;

Additional Key Words and Phrases: Shortcuts, keyboard, text editing, smartphone, gestures

ACM Reference format:
Huy Viet Le, Sven Mayer, Maximilian Weiß, Jonas Vogelsang, Henrike Weingärtner, and Niels Henze. 2020. Shortcut Gestures for Mobile Text Editing on Fully Touch Sensitive Smartphones. *ACM Trans. Comput.-Hum. Interact.* 27, 5, Article 33 (August 2020), 38 pages.
https://doi.org/10.1145/3396233

1 INTRODUCTION

Smartphones have replaced desktop computers for a wide range of everyday tasks. Mobile word processor [61], spreadsheets [59], and presentation programs [60] were installed over 1.5 billion times and became viable alternatives to their desktop counterparts. Recent advances in the field of mobile text entry [8, 74, 96] achieved improvements in typing speed which are almost comparable

Authors' addresses: H. V. Le, University of Stuttgart, Pfaffenwaldring 5a, 70569 Stuttgart, Baden-Württemberg, Germany; S. Mayer, Carnegie Mellon University, 5000 Forbes Ave, 15213 Pittsburgh, PA; M. Weiß, J. Vogelsang, and H. Weingärtner, University of Stuttgart, Pfaffenwaldring 5a, 70569 Stuttgart, Baden-Württemberg, Germany; N. Henze, University of Regensburg, Universitätsstraße 31, 93053 Regensburg, Bayern, Germany.

1073-0516/2020/08-ART33 $15.00
https://doi.org/10.1145/3396233

to those of hardware keyboards. Today, smartphones are used for a wide range of text-heavy activities [88] including e-mail writing, instant messaging, note taking, and even text editing in the cloud with over 500 million installations of GoogleDocs on Android devices [24]. While the mobility enables users to perform these tasks in a wide range of situations (e.g., while on the move), smartphones are compact devices which provide limited input and output capabilities compared to desktop computers. Hardware keyboards and mice provide over 100 different keys for text entry and shortcuts to frequently used functions. In contrast, touch input is limited to the two-dimensional (2D) coordinates provided by the screen to support the same tasks. The limited input capabilities of touchscreen-based devices pose severe usability challenges for mobile text editing. We need to improve their input capabilities to make text-heavy activities usable on mobile devices.

The fat-finger problem [7, 87] is a well-known challenge and makes precise caret placement and text selection frustrating. Further, the lack of shortcuts (e.g., hotkeys such as `Ctrl+C`) is a fundamental challenge for mobile interaction which even contradicts one of Shneiderman's golden rules for interface design, which asks to enable frequent users to use shortcuts [83]. Mobile operating systems often use dwell times (e.g., a long-press) and context menus to provide these shortcuts. However, these approaches are inconvenient and often slower than the keyboard counterparts. Previous work [22] and commercial products used on-screen gestures as shortcuts to frequently used functions. While this is a first step towards solving text editing challenges, touchscreens rely on direct manipulation where content and user interface (UI) elements can easily be selected and manipulated by touching them. Therefore, on-screen gestures can easily lead to overloading the UI with functions and thus unintended activations, interfere with other gesture-based UI functionalities (e.g., gesture keyboards), and still require a considerable effort to perform.

Previous work on *fully touch sensitive smartphones* (FTSP) showed that the back side, as well as the sides of a mobile device, can be used to perform input. Indeed, Le et al. [47] showed that over 70 % of the back side can be reached comfortably by all fingers, while further related work [48, 63] presented high-fidelity prototypes of FTSP. These devices enable the user to perform gestures with different fingers on the rear while the position of the fingers can be used as action modifiers. These types of phones would enable users to perform text editing shortcuts on the back and the sides nearly simultaneously to text-related activities on the touchscreen. In contrast to on-screen gestures, input on the back and the sides does not interrupt the input that a thumb or an index finger performs on the front touchscreen. This approach of providing shortcuts is similar to hotkeys on hardware keyboards and could soon be readily implemented on commodity devices, such as foldable smartphones.[1]

An important basis to design shortcuts for improving text editing on FTSPs is to understand which shortcuts are frequently needed and how they could be provided on FTSPs. In this article, we present the results of four user studies focusing on designing and evaluating text editing shortcuts.

2 STUDY OVERVIEW

Stationary computers with hardware keyboard and mouse are currently the primary device for text editing and thus suitable to study the used shortcuts. Thus, Study I analyses shortcuts performed by 15 expert users over 5 workdays in an in-the-wild study. Subsequent interviews revealed that major challenges for text-heavy activities on smartphones are limited input precision (e.g., placing the caret) and the lack of shortcuts. Despite these challenges, users still find text editing on smartphones essential. As these limitations can be resolved by offering more shortcuts (e.g., for navigating the caret), we conducted Study II to elicit gestures that provide shortcuts to the identified functions that are frequently used. We followed the formal methods proposed by

[1] https://www.techradar.com/news/samsung-galaxy-x-foldable-phone.

Wobbrock et al. [104] and focus on FTSPs to avoid conflicts with interacting with screen content and existing UI components such as gesture keyboards. To evaluate the elicited gesture set and its usability in realistic text-editing scenarios, we conducted two further studies to implement and evaluate a system that enables the use of the designed shortcuts. Thus, in Study III, we collected a ground truth dataset of participants performing the gestures on a FTSPs. The collected dataset is used to develop a deep learning model that infers shortcuts from capacitive images, two-dimensional matrices showing the raw measurements of a capacitive touchscreen. Study IV finally determines the accuracy of the trained model and evaluates the gesture set with realistic text editing scenarios to gather qualitative feedback on the gestures and their usability.

3 RELATED WORK

The goal of this work is to elicit, implement, and evaluate a set of gestures on a FTSP to support users with text-heavy activities. Thus, in the following, we review previous work on text editing using touchscreens, touch gestures, and input controls beyond the touchscreen.

3.1 Hotkeys for Graphical User Interfaces

Shortcuts are an essential part of graphical user interfaces (GUIs). The need for shortcuts has repeatedly been highlighted by previous work [25, 113]. Keyboard shortcuts or hotkeys are the de facto standard for current GUIs and clearly can provide performance advantages compared to menus and toolbars [25, 35, 43, 70]. While hotkeys are widely available for GUIs, adopting them is a major challenge for users. Lane et al. [43] showed that even expert users do not necessarily transition from the use of menus and toolbars to hotkeys. As identified by Scarr et al. [81], there is a dip in performance when transitioning from menus and toolbars to hotkeys. Accordingly, previous work on hotkeys focused on two aspects. To enable novice users to transition from the use of menus and toolbars, approaches have been developed which help novice users in the transition to expert users [25, 53]. Previous work also developed new approaches that enhance hotkeys typically to make them more accessible [79, 80, 113] or to enlarge the input space [15, 114]. While previous work showed that not even all expert users necessarily use hotkeys frequently [43], usage patterns of such users can still reveal those hotkeys that users consider important enough to make the transition to using them. At the same time, the challenge of adopting hotkeys for GUIs highlights the importance of considering shortcuts that are easy to guess from the very start.

3.2 Text Editing on Touchscreen Devices

Previous work presented a number of approaches to improve the input performance on mobile touchscreen devices. This includes simple key changes [8] to minimize the re-learning effort, optimized keyboard layouts [71, 75], reducing a systematic input bias [29, 30, 99], inserting frequent word chunks using surrounding menus [14, 34, 55], gesture keyboards [40, 111], and auto-corrections [96]. Novice typists can already reach a mean entry rate of 41 words per minute (WPM) [74, 96], which slowly becomes comparable with the mean entry rate of around 60 WPM on hardware keyboards [20, 39]. However, text editing on mobile touchscreen devices brings two main challenges. Precise caret positioning is difficult due to the fat-finger problem [7]. Further, the lack of shortcuts makes it difficult to perform a large number of text manipulation operations without going through long menus. In the following, we describe approaches from previous work to address these challenges.

3.2.1 Caret Placement and Text Selection. The fat-finger problem makes caret placement an inconvenient task. Previous work presented solutions that can be grouped into the following four

categories: (1) making the occluded area visible; (2) using a cursor; (3) performing input in (and thus occluding) unimportant areas; and (4) using shortcut commands.

Shift [97] is a technique to display the content occluded by the finger in a call-out above the finger. *Taptap* [76] shows an enlarged version of the occluded area to enable a precise selection. Apple iOS combines both techniques and shows the occluded display content using a magnifying lens to place the caret. Cursor-based approaches provide an offset cursor similar to an extendible thumb to avoid occlusion through fingers [37, 76, 82]. To avoid occlusion through touching less important areas, Suzuki et al. [89] proposed to move the text block instead of the cursor which stays in a fixed location. Similarly, previous work used Back-of-Device (BoD) interaction to select an object from the back to avoid occlusion [7, 100]. Shortcut commands are predominantly used by commercial keyboards (e.g., Swype [69], SwiftKey [91] and TouchPal [13]) which provide dedicated arrow keys. Closer to our work, Fuccella et al. [22] followed a similar approach and used gestures to navigate the caret. However, gestures can conflict with interacting with screen content and existing UI components such as gesture keyboards. There is no previous work that used BoD interaction to perform gestures for providing shortcuts to functions such as caret movement or clipboard management.

3.2.2 Shortcuts and Clipboard Management. Traditional hardware keyboards consist of 101–104 keys. Besides alphanumeric and punctuation keys, around a third of the keys do not generate any visible character themselves on the screen. Among others, these include navigation keys (e.g., arrow keys, `Home`, or `Page Down`), function keys (`F1`–`F12`), and modifier keys (e.g., `Shift`, `Ctrl`, and `Alt`). Modifier keys are the basis for shortcuts (e.g., `Ctrl+C` to copy) which are inevitable in leveraging expert performance [25]. This conforms with Nielsen's usability heuristics of providing the user with shortcuts [67]. Previous work has shown that shortcuts are faster than navigating through menus for the same function [58, 70].

Most touchscreen keyboards do not provide modifier keys so that well-known shortcuts from hardware keyboards are not available on mobile devices. Kristensson and Zhai [41] proposed command stroke gestures which are gesture traces from a modifier key (e.g., `Ctrl`) to the modified key (e.g., `C` for copy). The Swype keyboard implemented the command stroke concept to provide clipboard access through gestures. While this is a viable solution, it further consumes the device's limited screen space. Fuccella et al. [22] approached this challenge using gestures that are drawn on top of the on-screen keyboard to access the clipboard or move the caret using directional swipes. However, this concept conflicts with gesture keyboards while the command stroke concept requires users to learn the abstract gestures (e.g., ShapeWriter shortcuts on which command strokes are based) before gaining any profit in execution time [52, 110].

3.3 Touch Gestures on Mobile Devices

Previous work presented a broad range of use cases in which gestures are superior to traditional user interface components such as buttons. Li et al. [52] used letter-shaped gestures to perform search operations anywhere within an application. Poppinga et al. [72] investigated how people use letter-shaped gestures to launch applications. Commercial devices already offer a number of pre-defined simple gestures, including directional swipes, flinging, or simple shapes to launch applications. While these gestures are well-known and quick, their expressiveness is limited which can lead to association errors as shown by Nacenta et al. [66]. Therefore, approaches, such as OctoPocus [6] and ShadowGuides [21], to help users remembering touch gestures have been developed. In contrast to designer-defined gestures, user-defined gestures can also offer a better recall performance [66, 102]. Furthermore, they are also easier to perform and more appropriate than designer-defined gestures [65]. Thus, Wobbrock et al. [104] presented a methodology for

deriving user-defined gestures from users in elicitation studies. In these studies, users are shown the effect of gestures (referents) and are asked to come up with corresponding gestures that would cause the effect. Vatavu et al. [94, 95] presented formal measures including statistical tests to quantify guessability and agreement among the proposed gestures. This approach was applied in a wide range of use cases, including user-defined BoD gestures [86], mobile motion gestures [77], gestures for touch surfaces [104], and thus will be used in this work.

3.4 Input Controls Beyond the Touchscreen

Additional input controls can be used as shortcuts. Recent smartphones incorporate additional buttons and squeeze sensors to launch pre-defined functions (e.g., Bixby button and Edge Sense). Previous research investigated a wide range of use cases for input on the back and the side of mobile devices. Researchers attached pressure-sensitive sensors on the edges of a smartphone to enable interaction through pressure. For example, Wilson et al. [101] used pressure sensors to select menu items while Holman et al. [31] used the same sensors for scrolling. Previous work also used capacitive sensors on the edge to position UI components accordingly to the hand grip [12], rotate the screen [11], or to launch applications [10, 36]. Touch input on the back of the device enables a wide range of use cases, such as preventing shoulder surfing while authenticating [16], improving reachability by moving the front screen content [44], 3D object manipulation [4, 85], or performing user-defined gesture input [86], and zooming gestures in single-handed grips [62]. To understand interaction beyond the touchscreen, researchers investigated finger placement [51], movements [47, 50, 56], and on-device gestures [107] to propose guidelines for BoD interaction.

3.5 Summary

Previous work presented techniques that make text entry on smartphones almost as fast as on hardware keyboards [74, 96]. However, text-heavy activities are not widely adopted yet due to multiple challenges. Using direct touch to place the caret is inconvenient due to the fat-finger problem. When using hardware keyboards and GUIs, shortcuts in the form of hotkeys can clearly provide performance advantages compared to menus and toolbars [25, 35, 43, 70]. Such shortcuts are, however, widely unavailable on mobile touchscreen-based devices. Consequently, frequently used functions such as clipboard management and selecting a whole line are not accessible via shortcuts so that expert performances cannot be leveraged. Menus and toolbars are slower than shortcuts [25, 35, 43, 70]. They clutter the interface and require screen space which is especially scarce on mobile devices. Consequently, previous work proposed on-screen gestures as shortcuts. Touchscreens, however, rely on direct manipulation where content and UI elements can easily be selected and manipulated by touching them. On-screen gestures can also interfere with other gesture-based UI functionalities such as gesture keyboards. Thus, they are in conflict with touchscreen-based UIs.

In summary, the lack of useable shortcuts is a fundamental limitation of current mobile devices. As recent smartphones incorporate a wide range of input controls beyond the touchscreen, we focus on FTSPs and propose on-device gestures as shortcuts to frequently used functions. This avoids conflicts with the interaction with on-screen content and UI controls. It also avoids conflicts with existing gesture-based functionalities and extends the gesture input space through different gesture types (e.g., hand grip, pressure, and rear/side gestures). In four studies, we focus on eliciting, designing, and evaluating gestures for the use as shortcuts for text-heavy activities on FTSPs.

4 STUDY I: SHORTCUTS ON HARDWARE KEYBOARDS

While previous work already recorded the shortcuts used by expert users of word processors [1, 43], neither the specific keys and actions that are used in daily live nor how often specific

shortcuts are used have been reported. Furthermore, modern GUIs for word processing and programming changed substantially over the last decades. In addition, our work focuses on supporting not only word processing users but also programmers. Therefore, we conducted an in-the-wild study to analyze shortcuts performed on hardware keyboards by expert users. We investigate single and combination of keys which activate functions, and are not available on recent smartphone keyboards, e.g., standard on-screen keyboards for Android and iOS.

4.1 Apparatus

We used a low-level system-wide hook for keyboards to record shortcuts performed on hardware keyboard. Based on the *Java System Hook library*[2], we developed an application that runs in the background to log shortcuts into a text file. Shortcuts include all keys and key combinations that do not generate a new visible character on the screen, amongst others key combination that involves the modifier keys (`Ctrl`, `Win`, `Alt` as well as `Shift` for non-alphanumeric and non-punctuation keys), cursor keys, command keys (e.g., `Del` and `Insert`) and function keys (i.e., `F1–F12`, which are commonly unavailable on mobile on-screen keyboards). For each shortcut, we logged the foreground application's file name (e.g., *winword.exe*) and the timestamp. Our application automatically starts when the system boots. All participants were running our application on their work computer that runs Microsoft Windows.

4.2 Procedure and Participants

We first briefed participants on the collected data and their right to delete lines out of the log before the end of the study. After obtaining the participants' informed consent, we set up the application on their main work computer. Our application logged keyboard shortcuts over five full work days while weekend days and the starting day were excluded. After five work days, we invited the participants back to our lab to collect a copy of their logs and to conduct interviews. Specifically, we asked them about their experiences with text-heavy activities (i.e., word processing and programming) on hardware and touchscreen keyboards, as well as perceived advantages and disadvantages of both input modalities. In the form of a semi-structured interview, participants provided the comments orally and summarized them in written form afterward. This took around 20 minutes per participant.

We recruited 15 participants (8 male and 7 female) between the ages of 20 and 34 years (M = 25.5, SD = 4.2). Seven participants were research associates while eight participants were computer science students at a technical university located in central Europe. All participants were reportedly using their computer actively during the study. All participants stated that they write or edit text multiple times per day on hardware keyboards for tasks such as taking notes, writing reports and papers, and programming. Moreover, 12 participants stated that they write and edit text multiple times per day on smartphones to communicate with friends and family while two participants performed these tasks at least once per day.

4.3 Log Analysis: Shortcuts on Hardware Keyboards

We filtered all repetitions of the same shortcut within $1,000\,ms$ to avoid counting consecutively performed shortcuts multiple times (e.g., pressing `Alt+Tab` multiple times to select the desired window). Moreover, we removed all shortcut keys that were used only by a single participant to avoid user-specific shortcuts. After filtering, the dataset consisted of 67,943 shortcuts with 96 unique ones. On average, participants performed 787.5 (SD = 900.9) shortcuts per day of which 28.1 (SD = 13.4) were unique. During the 5 work days, participants performed shortcuts within 8.8 hours

[2]Java (low-level) System Hook library: http://github.com/kristian/system-hook/.

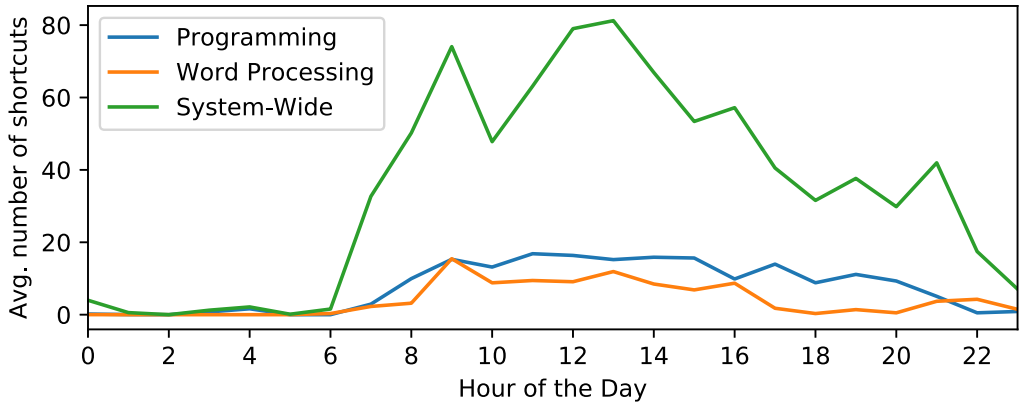


Fig. 1. Average number of shortcuts performed over the day per participant.

($SD = 3.2$) per day on average. Two researchers independently classified all logged applications into categories and discussed them afterward. Based on the discussion, we analyzed shortcuts within the following three categories: *word processing* which includes all applications focusing on editing and formatting text; *programming* which focus on coding tasks; and *system-wide* which includes all logged applications (e.g., system applications, browsers, and media player). Figure 1 visualizes the average use of shortcuts over a working day.

When categorizing the performed shortcuts by the number of keys, 43.0% were single-key shortcuts, 52.2% were double-key shortcuts, and 4.1% were triple-key shortcuts. Table 1 shows all shortcuts that were performed at least 1% in total averaged over all participants. To retrieve these results, we first calculated the percentage for each shortcut per participant and then averaged them over all participants. Overall, the arrow keys were the most used among all shortcut and application categories followed by clipboard management shortcuts (e.g., `Ctrl+V`) and the window switching shortcut (`Alt+TAB`). Shortcuts for caret movement and text selection were more frequently used in word processing applications than in programming environments and system-wide. In contrast, the `Left` and `Right` keys, as well as shortcuts for undo (`Ctrl+Z`), pasting (`Ctrl+C`), and saving (`Ctrl+S`) were performed more often in programming environments than system-wide on average. In general, 19 of the 24 most frequently used shortcuts benefit text-heavy activities (e.g., navigation, selecting, and clipboard). Three are widely available in text editors (undo, save, and search) while two are system-wide shortcuts (switching windows and open file manager).

4.4 Interviews: Hardware and Touchscreen Keyboards

One of the authors transcribed participants' answers from the interviews. Two researchers extracted arguments from the answers by splitting participants' phrases and printing them on paper cards. The two researchers then clustered the comments using affinity diagramming [28]. We focused on WORD PROCESSING and PROGRAMMING as two text-heavy activities and identified two main clusters which address the INPUT and the OUTPUT aspect. For each cluster, we found advantages and disadvantages for touchscreen and hardware keyboards. For the WORD PROCESSING task, participants made 46 comments addressing INPUT and 2 about OUTPUT challenges. Similarly, participants made 77 comments about INPUT and 12 about OUTPUT challenges for the PROGRAMMING task.

Word Processing. Two participants mentioned that the display size is one major concern that affects the OUTPUT. Specifically, "*the display is too small*" (P7) and thus "*lacks display size to fix*

Table 1. This Table Shows the Percentage of Occurrence of All Shortcuts That Represent at Least 1% in the Logs, Averaged Over All Participants for System-wide, Word Processing and Programming Applications

Shortcut	Function	All	Word P.	Progr.
	Single-Key Shortcuts			
DOWN	Move cursor down.	10.00	7.47	8.41
RIGHT	Move cursor right	8.79	10.20	12.09
LEFT	Move cursor left.	7.80	11.45	9.73
UP	Move cursor up.	6.56	5.79	5.55
DELETE	Delete character ahead.	2.60	4.29	2.44
END	Move cursor to end of line.	2.08	3.95	2.24
HOME	Move cursor to start of line.	1.17	1.17	1.07
	Double-Key Shortcuts			
Ctrl+V	Paste from clipboard.	9.18	9.90	11.20
Alt+TAB	Switch windows.	7.84	4.84	9.63
Ctrl+C	Copy to clipboard.	7.64	5.31	6.21
Ctrl+S	Save document.	4.82	9.29	13.48
Ctrl+A	Select all.	1.68	0.41	0.76
Win+E	Launch file manager.	1.61	1.37	0.19
Ctrl+X	Cut to clipboard.	1.61	1.66	1.78
Ctrl+F	Open Search.	1.05	0.98	0.55
Ctrl+T	New Tab.	1.02	0.09	0.03
Ctrl+Z	Undo last action.	0.87	0.86	1.53
Shift+HOME	Select until start of line.	0.85	0.70	1.85
Shift+RIGHT	Select character ahead.	0.72	1.09	0.70
Shift+LEFT	Select previous character.	0.63	1.47	0.89
Ctrl+LEFT	Move cursor to prev. word.	0.41	1.31	0.21
Ctrl+RIGHT	Move cursor to next word.	0.33	1.27	0.11
	Triple-Key Shortcuts			
Ctrl+Shift+LEFT	Select last word.	0.49	1.27	0.86
Ctrl+Shift+RIGHT	Select word ahead.	0.41	1.11	0.36

spelling mistakes while writing messages" (P9). Regarding INPUT, participants commented on the lack of haptics (25 comments) and shortcuts (21 comments). The lack of haptic feedback on touch-screen keyboards caused participants to be reportedly less accurate and thus slower ("*the large keys [on hardware keyboards] allow me to type faster*" – P3; "*[hardware keyboards] provide haptic feedback so that I do not have to look at the keyboard while typing*" – P7). Specifically, the lack of haptic feedback makes it difficult to perform precise operations such as placing the caret in a text (e.g., "*It is difficult to put the cursor to the desired text position.*" – P6; "*Choosing the position is inaccurate.*" – P4).

The latter challenge can be solved with shortcuts (e.g., Home, End, and arrow keys) to counteract the lack of precision. This conforms with statements from P2 who found that "*hardware keyboards offer many more options to change the caret positions*" and P12 who found that "*hardware keyboards are more precise since it is easy to switch between words and select whole lines.*" Moreover, "*[hardware keyboards] offer a lot of functions like select all, copy, paste, etc*" (P3). This makes text-heavy activities on hardware keyboards more convenient and faster ("*knowing the shortcuts makes [text editing]*

very fast" – P4) than on recent mobile devices ("*replacing existing text with copied text in Android at the right position is not really doable*" – P2). Shortcuts would circumvent the fat-finger problem to avoid imprecise selection, and save time since a long-press to access the clipboard can be replaced.

The statements indicate that hardware keyboards are superior due to higher precision and by providing more shortcuts. However, participants also stated that text-heavy activities on mobile devices are essential. Smartphones are mobile, which enables users to do a wide range of tasks while on the move. P13 stated that she "*[is] often doing things on the go and [does] not always have access to [her] laptop*" while P11 sees a clear advantage when "*[having] to send an email outside of the office, or to edit a document urgently over Dropbox or Google.*" Additionally, touchscreen keyboards comprise useful features that are not available on their physical counterpart. This includes auto correction ("*with auto correct, I do not always need to write the word completely*" – P4) and one-handed interaction which can be useful during secondary tasks ("*I find on-screen keyboards easier than a normal keyboard on a computer because I can type with just one finger on my phone*" – P8).

Programming. For PROGRAMMING tasks, we found 12 comments that address the OUTPUT. Compared to WORD PROCESSING, a "*large screen is required to analyze code in a comfortable way*" (P10) since "*code is often hundreds of lines long and can be barely interpreted, even on a normal-sized laptop screen. Viewing a large amount of text on a small screen would be annoying*" (P13). For challenges regarding INPUT, we found 77 comments that we clustered into three categories: typing speed, special characters, and shortcuts.

Conforming with comments for WORD PROCESSING, the typing speed on touchscreen keyboards is affected by the small display size (e.g., "*I cannot type as efficiently on an on-screen keyboard as on a hardware keyboard because I usually type with my two index fingers/thumbs instead of all ten fingers*" – P13). Moreover, special characters can only be accessed through switching layers or long-presses on most touchscreen keyboards due to the lack of modifier keys. This makes programming on touchscreens inconvenient since special characters are often required ("*I often need some specific characters which are only available by long-press*" – P7).

Participants predominantly mentioned the lack of shortcuts that make programming on hardware keyboards easier and faster ("*The use of shortcuts when using a hardware keyboard make coding tasks much easier and faster*" – P3). Specifically, shortcuts enable to navigate faster through structured code ("*functions to switch between methods (e.g., F3 in Eclipse)*" – P2) and text ("*Navigating text, which is often required when programming, is really hard on mobile devices.*" – P7). Moreover, programming environments enable users to create own shortcuts (e.g., for formatting and renaming) that make programming faster in general ("*You can use a lot of shortcuts and also create your own ones, what results in faster performance.*" – P10). Thus, the majority (10 participants) reportedly look up shortcuts on the internet (e.g., "*Looking up through search engines.*" – P13) while three participants reportedly look for shortcuts in menus. Despite the hardware keyboard's superiority, participants still find programming tasks essential in some situations (e.g., "*for emergency bug fixes*" – P1).

4.5 Discussion

We conducted an in-the-wild study with 15 expert users to analyze the usage of shortcuts on hardware keyboards. We further interviewed them afterward about their experiences in text-heavy activities on touchscreen and hardware keyboards. We found that the participants performed around 800 shortcuts on average per day and identified 24 unique shortcuts that they frequently used. Of these, 22 benefit text-heavy applications and enable one to select text, place the caret, access the clipboard, and activate helper functions (e.g., save and search). The majority of shortcuts are double-key shortcuts which are also available in the application menu.

The frequent usage of shortcuts indicates that users prefer them over buttons and menus in the user interface. This is further supported by participants who voluntarily look up shortcuts on the internet or try them out based on the menu description.

Interviews revealed that touchscreen keyboards are inferior to hardware keyboards due to the lack of precision (e.g., caret placement), shortcuts, and special characters for programming. Despite the disadvantages, participants highlighted that text-heavy activities are still essential on smartphones. Smartphones enable users to perform tasks in mobile situations (e.g., writing emails or doing emergency bug fixes while out of office). This conforms with the prominence of mobile alternatives of established computer programs (e.g., Word and Excel). Major challenges of touchscreen keyboards can be addressed by providing shortcuts to support mobile text editing. While the precision of caret placement can be increased with shortcuts known from hardware keyboards, shortcuts are necessary to provide a faster access to functions and special characters. Despite the necessity, recent touchscreen operating systems and keyboards do not provide shortcuts that enable a quick access to these functions. Instead, direct touch makes precise caret placement and text selection difficult while long presses for accessing the clipboard slows down the usage.

Solutions based on on-screen gestures are not applicable for a larger number of shortcuts due to interference with the interaction with screen content, existing UI components such as gesture keyboards as well as ambiguity errors in recognition. Instead, we propose to extend the gesture input space to the whole device surface on FTSPs. This extends the gesture input space which avoids interference and enables different gesture types (e.g., based on hand grip, pressure, and side-dependent gestures) that can be used similar to modifier keys on hardware keyboards. Since user-defined gestures are easier to perform and more appropriate than designer-defined gestures [65], we conducted a gesture elicitation study to derive a gesture set for improving text-heavy activities.

5 STUDY II: GESTURE ELICITATION

We conducted a study to elicit on-device gestures to provide frequently used shortcuts on FTSPs. Arguing for a user-centered perspective when designing gestures, Nielsen et al. proposed what has later become known as guessability studies or gesture elicitation [68]. A formal process for gesture elicitation studies has later been introduced by Wobbrock et al. [104].

Previous work argued that legacy bias (i.e., gesture proposals that are often biased by participants' experience with prior interfaces and technologies) is a major limitation of guessability studies [64, 78]. Ruiz and Vogel further argued that performance bias caused by artificial study settings that do not encourage consideration of long-term aspects leads to gestures sets that cause fatigue [78]. A range of variations [17], improvements [64, 78, 93], and alternatives [72, 106] have been showcased. Nonetheless, Vogiatzidakis and Koutsabasis, however, recently demonstrated that elicitation studies are the de-facto standard for designing gestures [98].

Gestures produced using elicitation studies are not only preferred by users [65] but are also easier to remember [66]. Therefore, we followed the formal method for gesture elicitation studies introduced by Wobbrock et al. [104] and used the AGreement Analysis Toolkit (AGATe) [94, 95] to analyze the collected gestures. This method uses a within-subjects design to ask participants for gesture proposals in a randomized order.

5.1 Referents

We derived 22 distinct referents as shown in Table 3 based on the frequently used shortcuts of the previous study. We considered all shortcuts, except application or system specific shortcuts that are not necessarily required on mobile devices. This includes the shortcut for application switching

Fig. 2. Setup for the elicitation study. We used the tablet for camera preview and the laptop to control the referents on the smartphone.

(`Alt+TAB`) for which mobile platforms already provide their own methods, and the save shortcut (`Ctrl+S`) for saving a document which a number of text editors already do automatically. The referents shown in Table 1 show the basic actions that are either related to caret placement, text selection, clipboard management, or document switching.

5.2 Apparatus and Procedure

To avoid distracting participants with an unfamiliar prototype or recognizer in the gesture elicitation process, we followed the concept of a *magic brick* [77] capable of detecting any gesture that was performed. Participants tested and demonstrated their proposed gestures on an off-the-shelf LG Nexus 5. Further, we used a custom Android application to show the referents using screenshots of the state before and after the action was performed. Displayed screenshots were controlled by an application on the experimenter's computer while input on the smartphone was disabled to avoid any reactions of the UI. All proposed gestures were recorded with a GoPro Hero 3 (audio and video) as shown in Figure 2.

After obtaining informed consent, we briefed participants on the procedure and collected demographic data including their experiences in using mobile devices for text editing. We instructed participants to think-aloud and explain the thought process as well as the proposed gesture. We briefed participants that gestures could be performed on the whole device surface with one or two hands. The order of the referents was randomized. In total, the study took around 45 minutes.

5.3 Participants

We recruited 18 participants (13 male and 5 female) between the ages of 20 and 34 ($M = 23.9$, $SD = 3.5$) who were staff or students at a technical university located in central Europe. None of the participants had participated in the previous study. All participants were right-handed with an

Table 2. Taxonomy of Gestures for Text Editing Shortcuts on a Full-touch Smartphone Based on Over 400 Elicited Gestures

Dimension	Category	Description
Nature	physical	Gesture acts physically on the object.
	symbolic	Gesture visually depicts a symbol.
	metaphorical	Gesture indicates a metaphor.
	abstract	Gesture-referent mapping is arbitrary.
Flow	discrete	Response occurs after the user acts.
	continuous	Response occurs while the user acts.
Complexity	simple	Gesture is atomic.
	compound	Gesture consists of atomic gestures.
Binding	object-centric	Location relative to object features.
	caret-centric	Location relative to caret features.
	mixed dep.	Any combination of above bindings.
	independent	Independent to any features.
Spatial	one-sided	Gesture performed on a single side.
	multi-sided	Gesture performed on multiple sides.

average hand size of 19.1 *cm* ($SD = 1.2$ *cm*) and used their smartphones multiple times per day. We reimbursed them with 5 EUR.

5.4 Results

Three researchers transcribed each proposed gesture, grouped them in case they were identical, and assigned them to the taxonomy shown in Table 2. Moreover, we transcribed the think-aloud protocols to analyze the thought process behind the proposed gestures. In total, participants performed 414 gestures from which we identified 138 unique gestures.

Taxonomy. We classified the transcribed gestures along the dimensions shown in Table 2. We adapted the taxonomy by Wobbrock et al. [104] originally proposed for touch surfaces and added the dimensions *complexity* and *spatial* while changing the *binding* dimension to match on-device gestures. This resulted in five dimensions: *nature*, *temporal*, *complexity*, *binding*, and *spatial*.

Nature describes the meaning of the gesture. *Physical* gestures directly manipulate objects (e.g., dragging the caret) while *symbolic* gestures depict an object (e.g., drawing scissors for cut). *Metaphorical* gestures manipulate imaginary objects (e.g., tracing a finger in a circle to simulate a scroll ring). We categorized gestures as *abstract* if the meaning was arbitrary (e.g., tapping three times to delete a selected word). *Flow* describes the visibility of a response; *discrete* if response is visible after a gesture was performed and *continuous* when response is shown while performing the gesture (e.g., scrolling on the right edge). The *Complexity* describes the composition of a gesture; *compound* if it comprises of atomic gestures and *atomic* if not. *Binding* describes the relation to the referent. An *object-centric* gesture refers to an object (e.g., a selected word) and *caret-centric* gestures refer to objects relative to the caret (e.g., delete previous word). *Independent* gestures do not refer to any object. *Spatiality* describes whether gestures are performed on a single side or on multiple sides simultaneously or in succession.

Categorization of Gestures. Figure 3 shows the taxonometric breakdown of the proposed gestures. Nearly half of the total gestures are physical gestures (44.9%). A large portion of the gestures are discrete (88.3%) and compound gestures (70.0%), whereas the majority refer to the

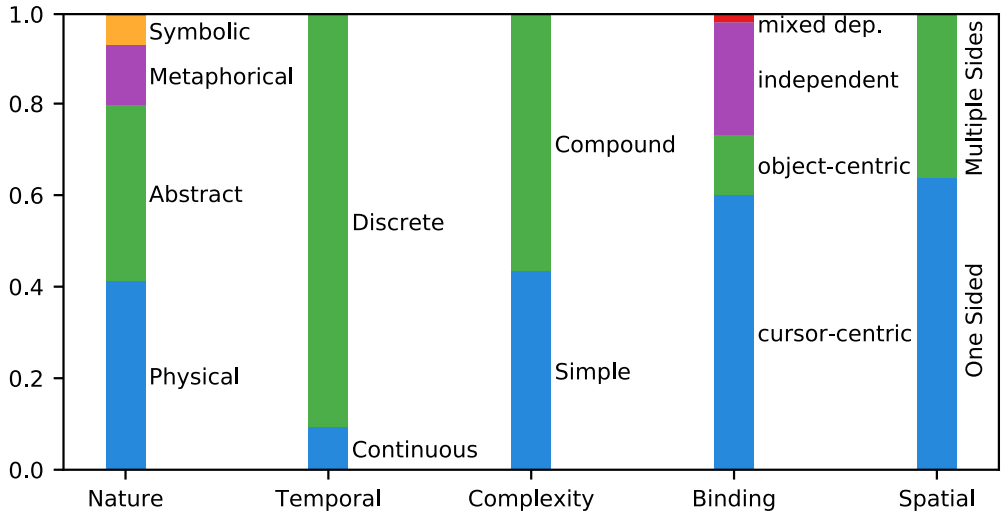


Fig. 3. Distribution of gestures in taxonomy categories as shown in Table 2.

caret's position (cursor-centric). Proposed gestures were performed on one side of the device (one-sided with 57.9%) and on multiple sides (41.7%).

Agreement Analysis. We used AGATe by Vatavu and Wobbrock [94, 95] to calculate the agreement and coagreement rates. The agreement rate $\mathcal{AR}$ describes the participants' consensus on the proposed gestures while the coagreement rate $\mathcal{CR}$ represents the agreement shared between two referents r_1 and r_2. The scores range between 0 and 1, whereas 1 represents the highest agreement. The overall agreement rate for the elicited gesture set is $\mathcal{AR} = 0.236$. We report the $\mathcal{AR}$ and $\mathcal{CR}$ for each referent in Table 3. We found a significant effect of referent type on agreement rate ($V_{rd(22,N=414)} = 314.798, p < 0.001$). Agreement rates for *caret positioning* referents are higher than the average with an $\mathcal{AR}$ of 0.359. We found a significant effect in this category ($V_{rd(7,N=144)} = 63.794, p < 0.001$). The agreement of *text selection* referents is $\mathcal{AR} = 0.214$ on average with a significant effect within the category ($V_{rd(8,N=162)} = 33.122, p < 0.001$). In contrast, the clipboard management category yielded a lower $\mathcal{AR}$ of 0.076 on average, while we found no significant agreement for all referents in this category ($V_{rd(2,N=54)} = 2.074, p = 1.000$). The group for moving the cursor yields the highest coagreement score with $\mathcal{CR} = 0.183$.

Mental Model Observations. All participants aimed to propose gestures that were consistent with each other. Consistency as an explanation was given for 12.6% of the proposed gestures whereas each participant mentioned it 5.1 times on average. We observed that participants tried to recall which gesture they had proposed for a similar referent and even asked which gesture they had used before. This was especially the case for gestures in the *caret positioning* and *text selection* category. Six participants (P2, P3, P6, P9, P12, and P14) tried to propose gestures that could be easily performed, especially when holding the device one-handed ("*[..] easy to do because index finger is already there*" – P2). Further, participants also explained that gestures were proposed to avoid unintended activations (e.g., "*more complex than just swiping so it doesn't happen accidentally*" - P1).

While the concept of on-device gestures was new to our participants, they (P1–P3, P6, P7, P10, P14, and P17) proposed gestures based on their previous technical experiences. For example, holding different positions on the rear was compared with modifier keys from keyboards ("*Pinkie and*

Table 3. Overview of Referents

Category	Referent	$\mathcal{AR}$	$\mathcal{MAR}$	$\mathcal{CR}$
Caret Positioning	Move Up	.523	.418	.183
	Move Down	.431		
	Move Left	.359		
	Move Right	.359		
	Move to previous word	.183	.245	.111
	Move to next word	.307		
	Move to start of line	.294	.285	.098
	Move to end of line	.275		
Text Selection	Select Up	.242	.275	.098
	Select Down	.235		
	Select Left	.379		
	Select Right	.242		
	Select left word	.235	.268	.137
	Select right word	.301		
	Select to start of line	.196	.219	.065
	Select to end of line	.242		
	Select All	.137	-	
Clipboard management	Copy	.059	.076	.000
	Cut	.072		
	Paste	.098		
Navigation	Switch to next document	.098	.098	.098
	Switch to previous document	.098		

$\mathcal{AR}$ represent the agreement score of each gesture while $\mathcal{MAR}$ represents the average agreement score for the respective group. $\mathcal{CR}$ represents the coagreement score.

Ring, are like Ctrl and shift" – P7, "*[..] holding like Ctrl button*" – P17). Further, we observed that participants changed the gesture they had in mind if they could not perform it with a sufficiently stable grip.

5.5 Gesture Set for Shortcuts in Text-Heavy Activities

The gesture set shown in Figure 4 consists of gestures with the highest agreement rate for each referent. Table 4 presents the relation of our final gesture set to the taxonomy discussed in Table 2. Moving the caret can be done with the index finger on the back to avoid occlusion issues (see Figure 4(a)). Participants envisioned the caret to move relative to the finger and suggested a movement threshold to avoid unintended movements during grip changes. Inspired by hardware keyboards, text selection can be done analogously to caret movement using the thumb as a modifier (*c.f.* `Ctrl` and arrow keys, see Figure 4(b) and 4(i)–4(l)). Moving the caret word-wise can be done by swiping the form of an arc to the respective direction (see Figure 4(g) and 4(h)). Participants explained this as a metaphorical leap over the previous/next word. Placing the caret at the start/end of the current line can be done with a double tap on the left/right side (see Figure 4(i) and 4(j)).

Selecting the whole text can be done with four taps on the front, while selected text can be copied by drawing the letter `C` on the back of the device (see Figure 4(n)). This refers to the shortcut on hardware keyboards (`Ctrl+C`). Participants cut text into the clipboard by swiping down two fingers on the rear (see Figure 4(o)) symbolizing a scissor, and pasting text using a double tap followed by a swipe down on the rear (see Figure 4(p)). To switch between tabs, participants proposed

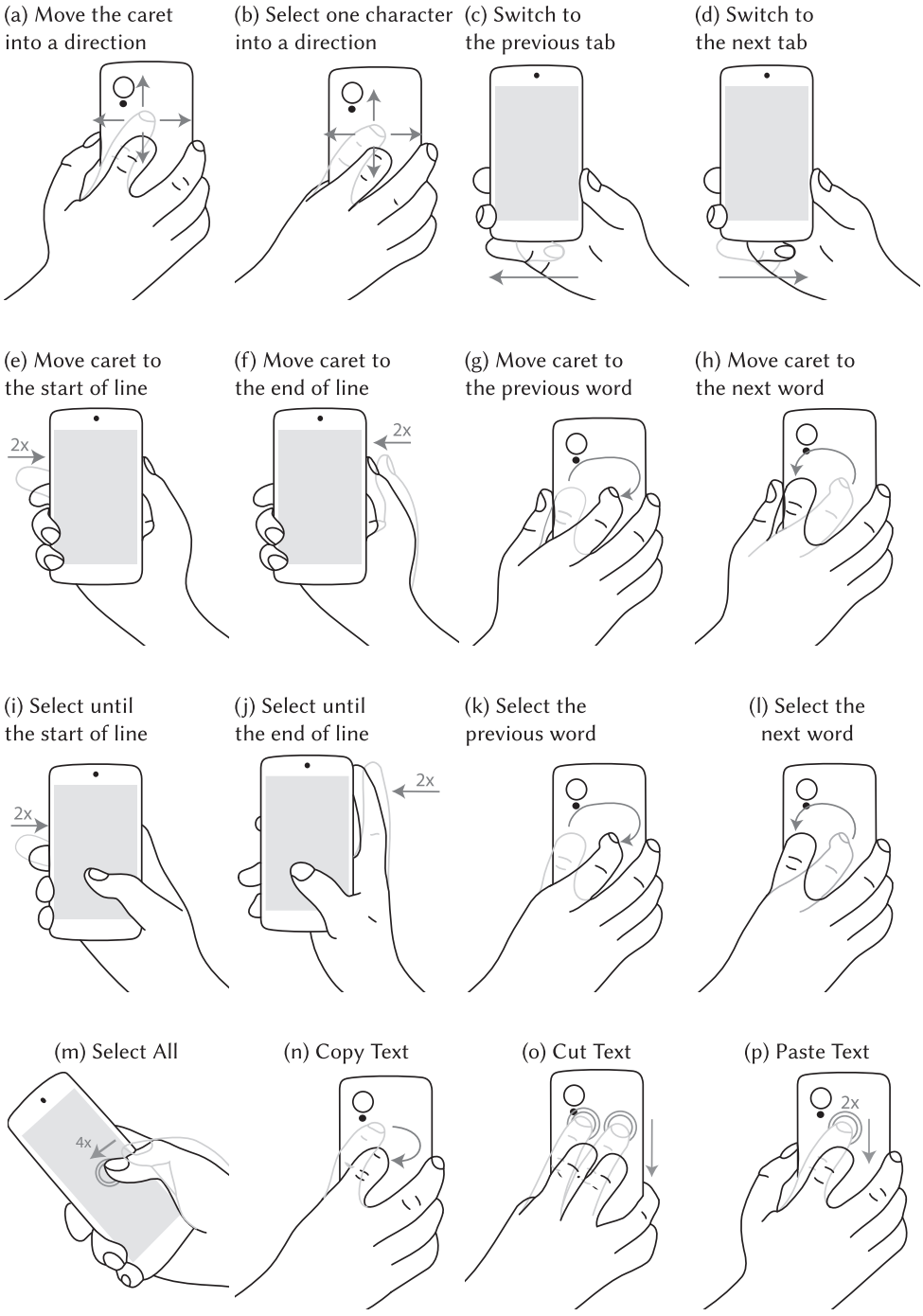


Fig. 4. The gesture set to provide shortcuts for word processing and programming activities. Shortcuts are performed on a smartphone that is capable of sensing touch input on the whole device surface. Gestures for moving and selecting the caret are summarized into one figure for all directions.

Table 4. Taxonomical Description of the Gesture Set Derived from the Results of the Gesture Elicitation Study

Referent	Nature	Flow	Complexity	Binding	Location	Spatial
Move Up Move Down Move Left Move Right	Physical	Discrete	Simple	Caret-centric	Relative	One-Sided
Move to previous word Move to next word	Symbolic	Discrete	Compound	Caret-centric	Relative	One-Sided
Move to start of line Move to end of line	Metaphorical	Discrete	Compound	Caret-centric	Relative	One-Sided
Select Up Select Down Select Left Select Right	Physical	Discrete	Compound	Caret-centric	Relative	Multi-Sided
Select left word Select right word	Symbolic	Discrete	Compound	Caret-centric	Relative	Multi-Sided
Select to start of line Select to end of line	Metaphorical	Discrete	Compound	Caret-centric	Absolute	Multi-Sided
Select All	Abstract	Discrete	Compound	Independent	Absolute	One-Sided
Copy Cut Paste	Symbolic	Discrete	Compound	Mixed dep.	Absolute	Multi-Sided
Switch to next doc. Switch to previous doc.	Symbolic	Discrete	Simple	Object-centric	Absolute	One-Sided

swiping left and right on the bottom edge which can be done with the little finger when used one-handed, and the thumb when used two-handed. Similar to the caret placement, the tab should not be switched before the swipe is performed for a minimum distance to avoid unintentional switches.

5.6 Discussion

We conducted a study to derive a gesture set that brings frequently used shortcuts from hardware keyboards to FTSPs. Figure 4 shows the gesture set that we discuss in the following. The gesture set achieves an overall agreement rate of 0.236 which is in line with gesture sets elicited in other domains [5, 92]. The agreement scores for *caret positioning* and *text selection* are higher than the average which could be due to the simplicity of the action that can be projected to a physical gesture (e.g., dragging the caret). In contrast, *clipboard management*, and *tab navigation* have lower agreement scores. This could be due to the abstractness of referents such as copying and cutting text that have no physical relations. Mobile operating systems often provide these functions through abstract buttons or pop-ups. Therefore, participants expressed these functions through symbolic gestures representing hardware keyboard shortcuts (e.g., `Ctrl+C`), or real-world objects such as scissors to cut, and convenient combinations of abstract taps and swipes. Thus, the results might be affected by legacy bias [64, 78]. While legacy bias prevents radically new gesture designs, it helps shorten the time and effort necessary to learn new ways of interaction [38]. Consequently,

there might still be room to further improve the gestures but the gesture set should be easy to learn and remember.

While participants assumed that the device is capable of detecting any performed gesture [77], recognizing the gesture set on a FTSP would comprise unintended activations, e.g., through grip changes. Participants considered this challenge and proposed compound gestures (e.g., tap then swipe) to counteract unintended activations. Moreover, the majority of the gestures are discrete so that the effect of activation is only shown after the gesture is fully performed. Thus, a series of distinct movements is required which makes unintended activations less likely. For continuous gestures (e.g., moving the caret), participants proposed to use movement thresholds to discriminate unintended movements (e.g., grip change) from intended movements. For example, the index finger on the back needs to move a minimum distance before the caret moves. Moreover, unintended activations could be further decreased by only accepting gestures when the expected number of fingers is moved (e.g., only the index finger is moving to position the caret).

6 STUDY III: IMPLEMENTING THE GESTURE SET ON A FULL-TOUCH SMARTPHONE

We implemented the elicited gesture set on a FTSP which we reproduced based on previous work by Le et al. [46, 48]. The prototype provides the raw capacitive measurements of the hand touching the device on all sides. This data enables us to recognize gestures performed on the device surface. While previous work used simple heuristics and gesture recognizers (e.g., "$1" [105]) to recognize 2D gestures performed on a flat touch surface, these approaches are not applicable to our use case. In particular, the raw capacitive measurements are low-resolution imprints of touches of all fingers instead of preprocessed 2D points as provided by touch controllers for common mobile operating systems. Therefore, we apply a deep learning-based approach to develop a model to recognize the gestures elicited in the previous study and shown in Figure 4. First, we invited a set of new participants which we instructed to perform our gesture set on the FTSP prototype. Second, we use the collected dataset consisting of capacitive images of touches to train a Long Short-Term Memory (LSTM) model which recognizes the gestures. In the following, we describe the study, model training and validation, as well as our mobile implementation.

6.1 Apparatus

We reproduced the FTSP prototype by Le et al. [48] which is shown in Figure 5. The prototype consists of a handheld device that senses capacitive touch input on its whole device surface (i.e., front, back, and the edges) with a resolution of $28 \times 32\,px$. The handheld device has a virtually identical size to a LG Nexus 5 to avoid influencing the usual grip of our participants. Capacitive measurements are provided to the application layer every $50\,ms$. For more technical details, we refer to the detailed description by Le et al. [48].

We developed a custom application which collects the capacitive images in the background while the participants followed the study instructions. During the study, participants were seated on a chair without armrests in front of a screen that shows the instructions. The instructions show gestures that participants are supposed to perform. Moreover, participants rated the gestures using a mouse after performing them on the FTSP.

6.2 Participants

We recruited 28 participants (21 male and 7 female) between the ages of 20 and 29 ($M = 23.7$, $SD = 3.4$) which did not participate in the previous studies. Two participants were left-handed, and 26 were right-handed. The average hand size was measured from the wrist crease to the middle fingertip and ranged from $16.0\,cm$ to $25.0\,cm$ ($M = 19.3\,cm$, $SD = 1.6\,cm$). With this, our set of participants include samples of the 5th and 95th percentile of the anthropometric data reported

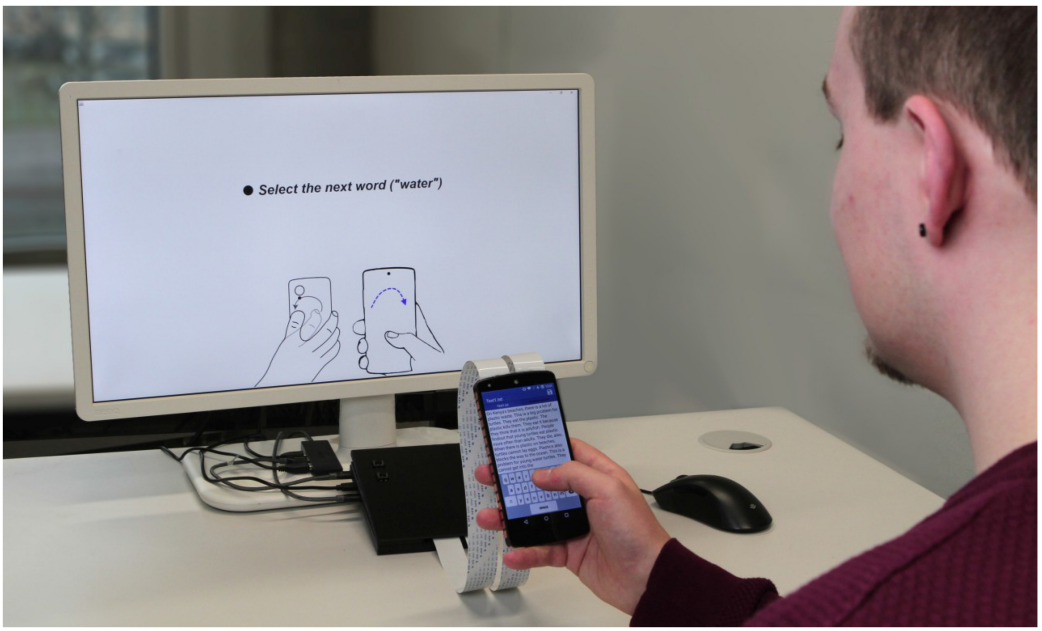


Fig. 5. This figure shows the apparatus of the study. A participant holds our smartphone prototype to edit a text based on instructions shown on the screen in front of the participant.

in previous work [73]. Thus, the sample can be considered as representative. Participants were reimbursed with 10 EUR for their participation.

6.3 Procedure and Study Design

After obtaining informed consent, we asked participants to fill out a demographics questionnaire and measured their hand size. We then briefed them about the smartphone prototype and the purpose of the study. The study consists of six phases in which each of the gesture has to be performed by the participant. Using a custom application, the experimenter started and stopped the recording while instructing participants when to perform the gesture.

In the first phase, the experimenter instructed participants to perform the gesture as shown on the display in front of them. In case it was unclear, the experimenter further demonstrated the gesture on another smartphone. Before starting the recording of the respective gesture, participants were asked to perform the gesture on trial to ensure that everything was understood. In the second phase, participants were explained the function of each gesture before they were recorded. The display in front of them shows the function name while the smartphone exemplarily shows the before and after states of a text editor. Moreover, the experimenter explained each gesture orally and provided examples in case participants did not fully understood the function. After everything was understood, participants were asked to perform the gesture as if they are currently performing text editing tasks.

In the remaining four phases, we instructed participants to perform the gesture as shown on the display while the smartphone showed exemplary before and after states. These phases are used for collecting more data from the participants. In total, participants performed each gesture 6 times, which results in $28 \times 6 = 168$ samples per gesture per participant.

6.4 Modeling

We describe our preprocessing and training steps to train a user-independent model to recognize shortcut gestures.

Dataset & Pre-Processing. We collected 477,605 capacitive images in total throughout the study. Our preprocessing steps includes synchronizing the capacitive images from all sides, cleaning the dataset, and preparing the samples for a LSTM model. We performed the following steps:

(1) *Matching capacitive images from all sides:* Based on the timestamps of the capacitive images, we merged the capacitive images of all sides into a combined one with a size of $32 \times 28\,px$. To reduce the processor workload of a mobile deployment (i.e., the average number of capacitive images within one gesture), we merged the front image with the back image and the closest side sensor image. We used the closest side sensor image (timestamp-wise) since the side sensors have a higher sampling rate.
(2) *Dataset Cleaning:* For each gesture, we removed all frames with no movements (i.e., frames in which the participant only held the device to wait for the next task). This was done by determining blobs in the capacitive image and an element-wise equal operation[3] within a tolerance of $4.1\,mm$ (equals to the size of a pixel in the capacitive image). We further removed all gestures from our dataset which did not include any movements to prevent errors.
(3) *Summarizing Gestures*: Since the input on the front touchscreen is limited to either a thumb placement as modifier (i.e., switching from moving cursor gestures to selecting gestures) or the 4x tap gesture for the select all action, we omitted the front data and summarized the gestures accordingly based on their back and side data. In particular, we summarized gestures (a–e) with (f–j) into the five respective classes and differentiate between selection and movement during run time based on the touch API from Android (i.e., whether a touch is registered on the front or not). Moreover, we removed the gestures b, c, g, and h since a simple double tap on the sides can be easily recognized manually. We ended up with 12 classes with the resulting capacitive images being $17 \times 28\,px$.
(4) *Preparing samples for LSTM training*: Since LSTM models require a fixed input size (i.e., equal number of timesteps per gesture), we padded and trimmed gestures respectively to a sample size of 20 based on the average length of a gesture in our dataset ($M = 16.04$, $SD = 5.7$).

In total, after preprocessing our dataset consists of 114,720 capacitive images.

Gesture Production Time. To ensure that gestures can be performed fast and effective, we first calculate the production times based on the preprocessed data. We measured the time from starting to produce a gesture to finishing it. The average time to produce a gesture was $1.28\,sec$ ($SD = 0.73$). The slowest gesture was the "Select previous word" gesture with $2.02\,sec$ ($SD = 0.9$). On the other hand, the fastest gesture was the "Move left" gesture with $0.74\,sec$ ($SD = 0.57$). For a full overview see Figure 6. Overall the production time of the back of device gestures are in line with production times reported by previous work [108]. Wolf et al. [108] asked participants to perform different touch gestures on a tablet five times in a row. They found durations between 0.2 and 1.1 seconds. While not directly comparable, gesture production times are similar to results reported for hotkeys on desktop computers and potentially shorter compared to the use of menus and toolbars. Lane et al. [43] asked participants to activate four shortcuts using hotkeys, icons or a menu. After

[3]We used numpy's allclose operation: `abs(a-b) <= (atol + rtol * abs(b))`.

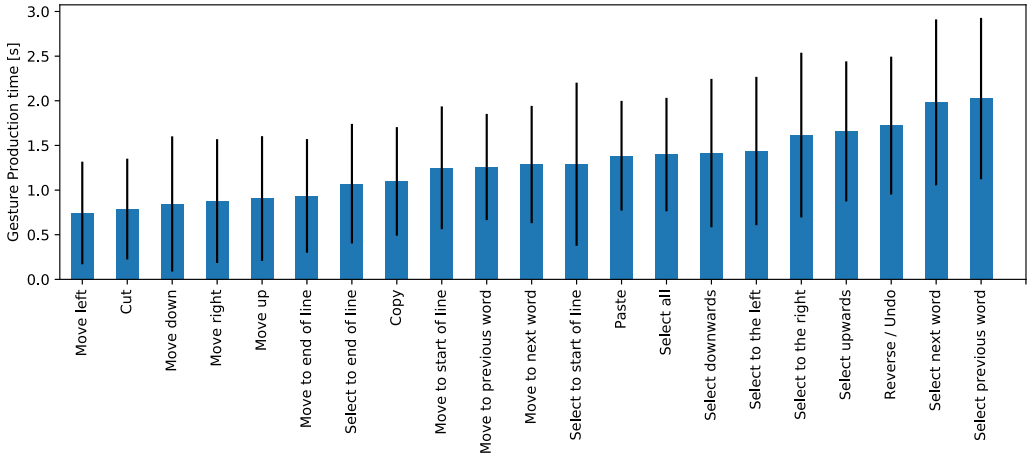


Fig. 6. Gesture production time in seconds for each of the different shortcut gestures. The error bars represent the standard deviation.

training, hotkeys required 1.4 s, icons 2.2 s and the menu 3.1 s to activate a function [43]. It is important to stress, however, that production time we measured and the time measured by Lane et al. [43] are not directly comparable.

Model Architecture & Training. To train the model, we used a participant-wise split of 80 %:20 % for training and testing, i.e., we trained the model on data from 22 participants and tested the model on the remaining 6 participants. We did not use a validation set as we determine the validation accuracy in the next study described in Section 7.

We implemented a convolutional neural network (CNN)-LSTM [18, 19] using *Keras* 2.1.3 based on the *TensorFlow* backend. While we experimented with convolutional LSTMs [109, 112] (a fully connected LSTM with convolutional input and recurrent transformations) in Keras[4] and stateful LSTMs with continuous capacitive image input, we found that a CNN-LSTM leads to the highest classification accuracy. We performed a grid search as proposed by Hsu et al. [32] to determine the most suitable network architecture and hyperparameters. If we do not report a hyperparameter in the following, we applied the standard value (e.g., optimizer settings) as reported in Keras' documentation.

The architecture of our final CNN-LSTM is shown in Figure 7. We adapted the convolution layers which worked best for previous work that used capacitive images [48] and used a `TimeDistributed`[5] layer to attach it to the LSTM part; a grid search focusing on the number of filters and kernel size in the convolution layer did not reveal any improvements in accuracy. After a further grid search on the LSTM part, we found that 100 LSTM units and a dropout factor of 0.7 achieved the highest accuracy. With this model, we further experimented with different number of timesteps (i.e., capacitive images) per gesture by increasing/decreasing them in steps of 5. We found that a window size of 20 (four more than the average) yielded the highest performance.

Similar to Le et al. [48], we trained the CNN-LSTM with an RMSprop optimizer [90] but with a batch size of 16. Further, we determined 0.001 to be the most suitable initial learning rate after testing in steps of negative powers of 10. We experimented with batch normalization [33] and L2 Regularization, but did not find any improvements in the overall performance.

[4]Convolutional LSTMs are provided as ConvLSTM2D in Keras: https://keras.io/layers/recurrent/#ConvLSTM2D.
[5]`TimeDistributed` layer in *Keras*: https://keras.io/layers/wrappers/#TimeDistributed.

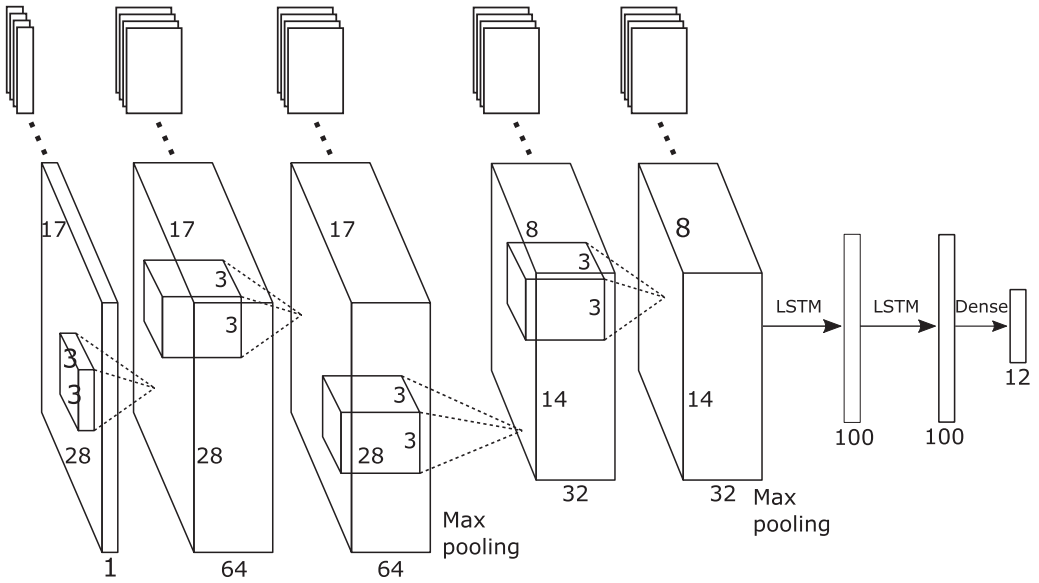


Fig. 7. An illustration of the architecture of our CNN-LSTM model which we used to classify gestures on the FTSP. The first four layers are convolution layers wrapped in a `TimeDistributed` layer whose output is then fed into two subsequent LSTM layers with 100 units each. The output are 12 values representing the probabilities for each gesture class.

Model Accuracy. Based on the test set, our model identifies the 12 gesture classes with an accuracy of 80.92 %, precision of 79.32 %, and recall of 78.28 %. Figure 8 shows the confusion matrix.

6.5 Mobile Implementation

We used *TensorFlow Mobile*[6] for Android on the FTSP's processing unit responsible for the front display to run the CNN that estimates the fingertip positions. Capacitive images from the back side and the edges are sent to the front device that merges the data into an input matrix. The input consists of a 17×28 8-bit image representing the back and edges. A model inference for one capacitive image takes 163.7 *ms* on average ($SD = 34.4\,ms$, $min = 101\,ms$, $max = 252\,ms$) over 1,000 runs on our prototype. While we exported the model without any modifications, the inference time can be reduced significantly with optimization techniques such as quantization [27] and pruning [3] for a small loss in accuracy, or using recent processors which are optimized for neural networks[7] (e.g., Snapdragon 845).

7 STUDY IV: EVALUATION OF SHORTCUT GESTURES

We conducted a final study in which we (i) determine the accuracy of the trained model with a new set of participants which were not involved in the model development process, (ii) collect qualitative feedback about our gesture set in a Wizard-of-Oz setting which excludes the effect of potential recognition errors, and (iii) collect qualitative feedback about the perceived usability of our prototype including the gesture recognizer, FTSP, and the text-editing functions.

[6]TensorFlow Mobile website: www.tensorflow.org/mobile/.
[7]www.qualcomm.com/snapdragon/artificial-intelligence.

Fig. 8. Confusion matrix of the CNN-LSTM for identifying 12 gesture classes with an accuracy of 80.92 %. The values shown in the figure represent the accuracy in percent (%). The *x*-axis represents the predicted class while the *y*-axis represents the actual class.

7.1 Study Procedure and Design

We designed three tasks to evaluate the three aspects described above. Prior to the study, we obtained informed consent, measured the participants' hand sizes, and handed them an instruction sheet which explains all parts of the study. Participants could refer to the instruction sheet at any time during the study.

Part 1: Accuracy Validation and Introduction of Gestures. This part is similar to the data collection study as described in Section 6. The focus is on obtaining a validation set with new participants while introducing and explaining the shortcuts for the next parts of the study. The experimenter explained each gesture to the participant using an image of the gesture displayed on the screen in front of the participant (see Figure 5), and by demonstrating it on another smartphone. Moreover, the experimenter explains the function of the gesture which is supported by an exemplary before and after state image shown on the FTSP.

Part 2: Wizard-of-Oz Evaluation of the Gesture Set. Based on a Wizard-of-Oz implementation of our gesture recognizer, we instructed participants to perform a set of pre-defined text-editing tasks in our application. This part uses a Wizard-of-Oz implementation to avoid influencing participants with potentially wrong recognitions of our model. Using a 2×3 within-subjects design, we compare accessing functions for caret movement, text selection, and clipboard management of Android's recent methods (STANDARD) with our gesture set (GESTURE). As different font sizes affect the input precision, we used three font sizes from previous work by Fuccella et al. [22] to compare the two input methods: 1.75 *mm*, 3.25 *mm*, and 4.75 *mm*. The conditions were counterbalanced with a Latin square and used six different texts to avoid learning effects.

Participants performed a set of text-editing operations which were pre-ordered and described on the screen in front of the participant. Due to the fixed order, a trained experimenter could carefully observe the performed input of the participant to trigger the respective action using a

Wizard-of-Oz controller on another smartphone. With the STANDARD method, the cursor can be moved by touching at the desired location. Text selection works similar with a long-press prior to the selection which also displays the clipboard functions in a top menu (i.e., ActionBar).

After each condition, we collected qualitative feedback in the form of a System Usability Scale [9], and the seven-point Likert scale questions adapted from previous work [45]. At the end of this part, we further conducted a semi-structured interview in which we focused on comparing the conditions.

Part 3: Gesture Set Evaluation based on the Model. This part focuses on collecting qualitative feedback on the perceived usability of our gesture classifier for a new text with a font size of 3.25 *mm*. Similar to the previous step, participants were given a set of text editing operations which they performed with our gesture set. After the task, we conducted a semi-structured interview in which we focus on the perceived usability and usefulness of our gesture set for text editing tasks.

7.2 Apparatus

We developed a custom text-editor application which provides the required functions for caret movement, selection, and clipboard management. For the use case evaluations, we included seven simple texts (six for the second part; one for the third part) from an English learning website[8] which are editable with an on-screen keyboard, as well as shortcut gestures or Android's standard text-editing methods. The shortcuts can be activated either with a remote Android application on another smartphone (Wizard-of-Oz controller for Part 2), or by performing the shortcut gestures on the FTSP (Part 3). The Android application deployed on the back unit performs the gesture classification based on its own and the capacitive images from the sides and sends the classification result to the text editor application on the front unit.

Since gestures which require a thumb placement on the front touchscreen would interfere with the long-press mechanism of most Android keyboards (e.g., punctuation marks on the built-in keyboard or GBoard), we developed a custom keyboard which detects long-presses and uses them to differentiate between caret movement gestures (no thumb) and selection gestures (thumb on the front) instead of inserting punctuation marks. The keyboard further changes its background color to indicate the selection mode.

7.3 Participants

We recruited 12 participants (6 female and 6 male) between the ages of 17 and 25 ($M = 21.1$, $SD = 2.7$). These participants neither participate in Study II nor in Study III. All participants were right-handed (one was ambidextrous) and reportedly use mobile touch-based devices multiple times per day. Participants were reimbursed with a credit point for their lecture.

7.4 Results

We present the results of each part of the user study. The validation accuracy as well as the ratings on easiness and suitability are result of the first part. Subjective ratings and the feedback gathered in the semi-structured interview are collected after the second part. The implementation which we evaluated in the third part was evaluated with semi-structured interviews and a questionnaire focusing on whether the participants would prefer our implementation or the recent text editing features integrated into Android. In total, the results describe the usefulness and usability of the gesture set, its use cases, and our prototypical implementation.

[8] https://www.newsinlevels.com/.

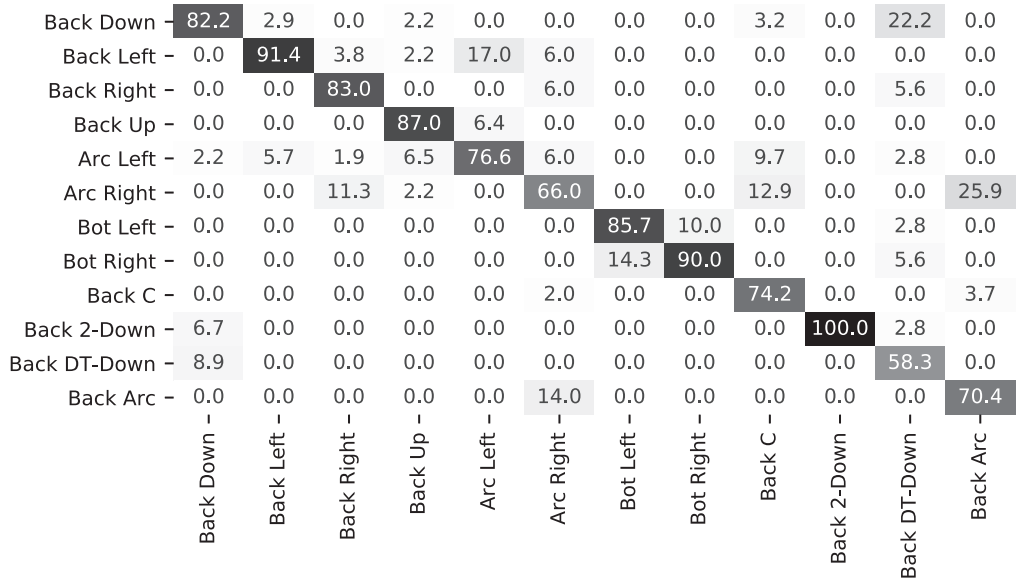


Fig. 9. Confusion matrix describing the validation accuracy of the CNN-LSTM presented in Section 6.4. The values represent the accuracy in percent (%). The *x*-axis represents the predicted class while the *y*-axis represents the actual class.

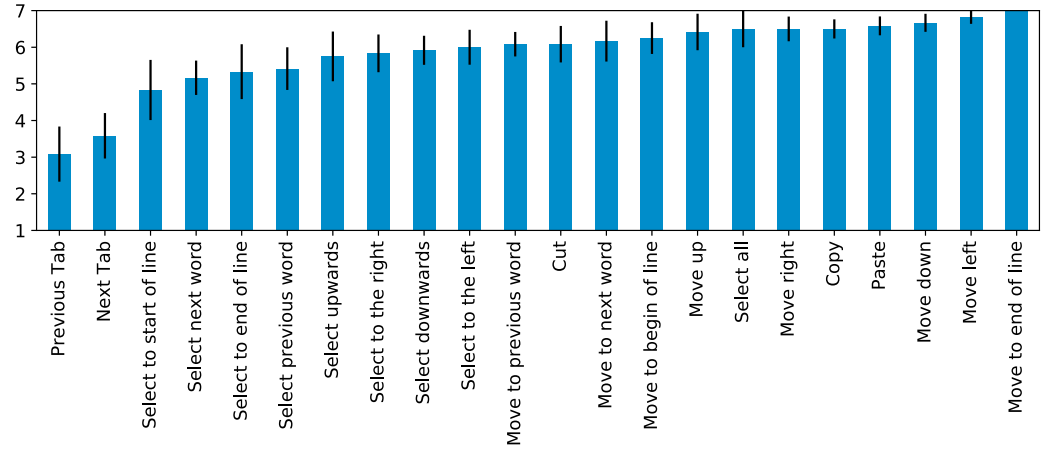
Validation Accuracy. Based on the capacitive images of gestures performed by the new set of participants, our model achieved a mean accuracy of 79.19 % (SD = 8.21 %, min = 63.64 %, max = 89.66 %). The mean precision is 80.88 % (SD = 8.27 %, min = 62.22 %, max = 95.0 %) while the recall is 77.67 % (SD = 7.05 %, min = 62.5 %, max = 88.89 %). The confusion matrix for this validation is presented in Figure 9.

Easiness and suitability of each gesture. Adapting the approach from previous work by Wobbrock et al. [104], we asked participants to rate the ease (i.e., the gesture is easy to perform) and goodness (i.e., the gesture is a good match for its intended purpose) of each gesture and its assigned function on a seven-point Likert scale. The results are shown in Figure 10. The average rating for ease is 5.84 (SD = .91, min = 3.31, max = 7.0), while the average goodness is 6.06 (SD = .57, min = 4.77, max = 6.85).

Subjective Ratings. For each of the four gesture categories (i.e., placing, selecting, clipboard access, and tab switching), we collected subjective ratings in the form of seven-point Likert scale questions and semi-structured interviews. These are used to compare text editing using our gesture set with text-editing operations as implemented in recent Android systems. Figure 11 shows the results. For each gesture category, we conducted two-way analysis of variances (ANOVAs) on the five ratings on which we applied the Aligned Rank Transform procedure using the ARTool [103] to align and rank the data.

For the *caret placing* gestures, we found significant main effects for SIZE and METHOD for the properties *easiness*, *speed*, *comfort*, and *effort* ($p < 0.01$) while we found a significant two-way interaction effect between SIZE × METHOD for the properties *easiness* and *effort* ($p < 0.05$). For the *text selecting* gestures, we found significant main effects for SIZE for the *easiness* property ($p = 0.01$), for METHOD for all properties ($p < 0.01$), and a significant two-way interaction effect between SIZE × METHOD for the properties *easiness* and *comfort* ($p < 0.05$). For the *clipboard access* gestures, we

(a) Ease rating ("The gesture is easy to perform").



(b) Goodness rating ("The gesture is a good match for its intended purpose").

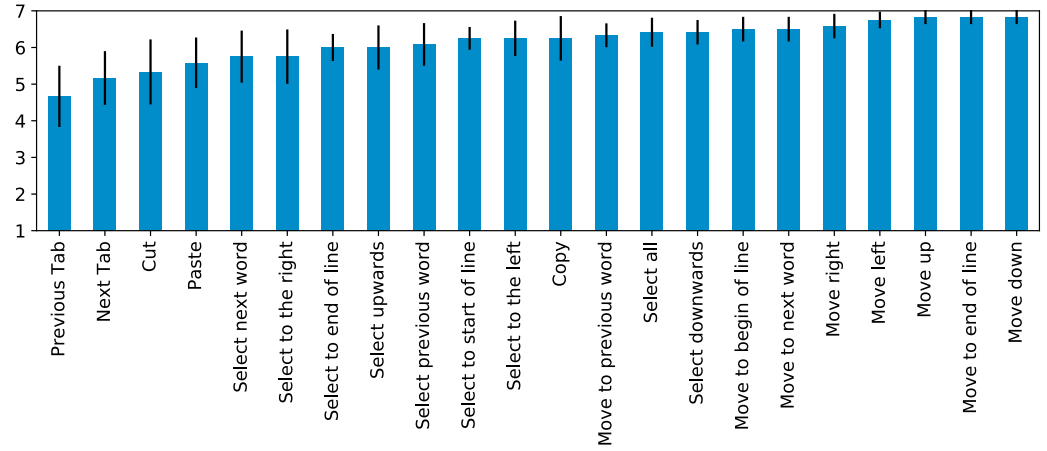


Fig. 10. Participants' ratings on a seven-point Likert scale for the (a) easiness and (b) goodness of each gesture.

found significant main effects for METHOD for the properties *easiness* and *suitability* ($p < 0.05$). For the *tab switching* gestures, we found significant main effects for METHOD for all properties ($p < 0.001$). The tests did not reveal any other significant effects besides the ones described above. Table 5 provides an overview of all F and p-values of the two-way ANOVA tests while Figure 11 shows the rating means and standard deviations.

Semi-Structured Interviews. We interviewed the participants after they performed the text-editing tasks with our gesture set and Android's standard text-editing mechanisms. Two researchers employed a simplified version of qualitative coding with affinity diagramming [28] to analyze the results. This includes transcribing the interviews, extracting the arguments from the participants' answers, printing them on paper cards, and finally clustering the answers. In the following, we first present general impressions of text editing with gestures (including advantages,

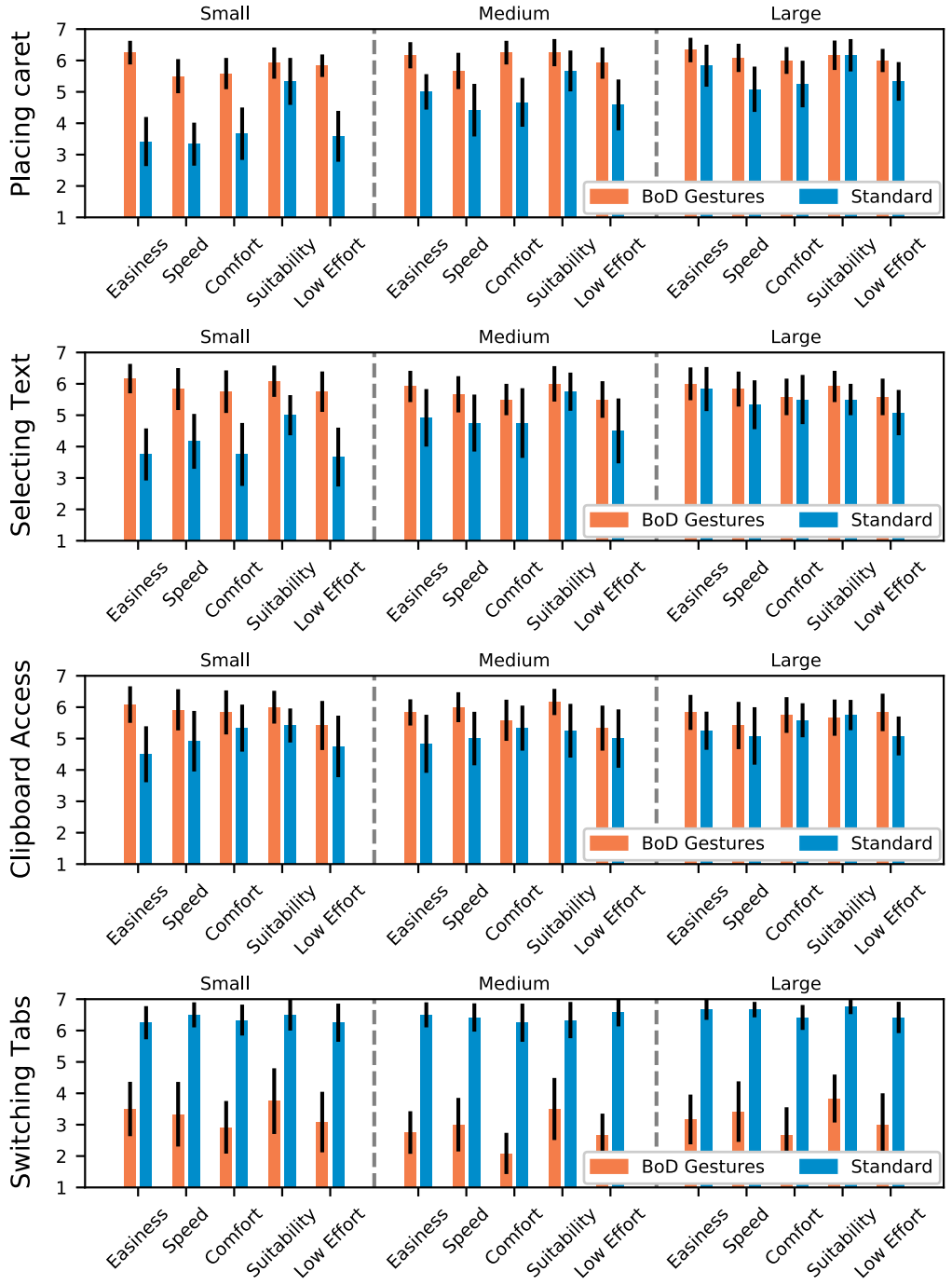


Fig. 11. Subjective ratings after editing a text with standard touch input and our BoD gestures.

Table 5. Results of the Two-way ANOVAs for Each Gesture Class and Property

Variable	Easiness		Speed		Comfort		Suitability		Effort	
	F	p	F	p	F	p	F	p	F	p
SIZE	13.6	<.001	7.0	.002	6.6	.003	2.0	.147	5.6	.006
METHOD	62.9	<.001	3.4	<.001	32.4	<.001	1.4	.248	33.0	<.001
S×M	13.2	<.001	2.0	.1517	2.3	.138	.7	.522	3.8	.027
SIZE	4.9	.01	1.3	.28	2.2	.127	.9	.40	1.4	.246
METHOD	18.0	<.001	9.0	.004	4.7	.035	8.9	.004	14.5	<.001
S×M	6.5	<.001	1.6	.214	3.8	.027	1.2	.322	2.8	.07
SIZE	.3	.757	.1	.939	.2	.785	.1	.938	.1	.937
METHOD	14.2	<.001	2.1	.155	2.7	.105	4.6	.035	3.0	.087
S×M	1.1	.331	.7	.506	.2	.813	1.2	.318	.3	.764
SIZE	1.1	.327	.9	.413	1.1	.333	.4	.655	.4	.656
METHOD	189.9	<.001	167.2	<.001	189.2	<.001	111.1	<.001	174.0	<.001
S×M	1.8	.186	.3	.711	.8	.472	.1	.883	1.6	.217

The row blocks refer to the gesture classes, from top to bottom: caret placement, text selection, clipboard management, and tab switching. S × M refer to the two-way interaction effect between SIZE × METHOD.

disadvantages, and neutral comments) and then present the feedback for each gesture group separately.

General Impressions. When asked about the first impression of our text-editing gestures, participants made 34 positive, 15 neutral, and 7 negative comments. All participants mentioned at least one positive aspect for text editing with our gestures. In particular, participants preferred our gestures over Android's standard text-editing mechanisms due to the following reasons. First, participants preferred the gestures to improve the usability and input pace with fingers on the back which were previously unused ("*I really like the idea, since we have many unused fingers on the back*" – P13; "*Some gestures improve the speed of use, especially if the user gets used to it*" – P11). Further, participants recognized that BoD gestures improve the input precision. Among others, P8 found that "*[BoD gestures] makes placing the cursor much easier*" and "*more precise*" (P2) which "*is much better especially for people with fat fingers*" (P12). While participants mentioned that the usability of Android's standard text-editing mechanisms highly depends on the text size ("*the smaller the text is, the more you had to fiddle around*" – P12; "*if the text is very large, the normal version worked fine*" – P5), the usability of the our gesture set is independent from sizes of text, fingers, and hands ("*For me, the gestures were better for all text sizes*" – P4, "*an advantage is that, independent from the size of the hand and fingers, you can work with it relatively well*" – P10).

Participants also made comments which are positive but includes minor concerns. In particular, participants "*found some of the gestures difficult [to perform]*" (P3), however "*normal touch is still [perceived as] worse*" (P3). Since some gestures were perceived as difficult (e.g., "*I found the gesture for tab switching intuitive, but it was difficult to perform*" – P9), participants envisioned that they "*would use a mixture of both methods for text editing*" (P5, P8, P12). Regarding the FTSP prototype, participants also found that "*the glass [of the back touchscreen] induces friction*" (P7) which makes performing gestures uncomfortable. This could be readily improved in a market-ready version.

Negative comments focus on the extended input space and indicate two challenges which concerned the participants. First, they assumed that devices could become more expensive and fragile due to the additional touch sensors ("*the device could become much more complex with more sensors which could break*" – P13). Second, regarding the ergonomics, participants assumed that disadvantages include that more fingers need to be moved ("*[the user] needs all five fingers*" – P1; "*I need*

to move my fingers a lot" – P6) and grip stability challenges for users which already have trouble holding a device in a single-handed grip ("*with small hands, some gestures are difficult to enter while holding the device*" – P10).

Impressions on Cursor Placing Gestures. When asked about impressions on the cursor placing gestures, we received 10 positive, 1 neutral and 1 negative comment. All except P7 and P11 provided positive comments which describe the placing gestures as "*intuitive*" (P9), "*(very) helpful*" (P5 and P8), and well working (P1, P2, P3, P4, P6, P9, P10, and P12). P12 explained that "*the gestures on the back are much better, because [she] often moves the cursor beyond the target with normal touch because of [her] fat finger].*" In contrast to placing the cursor with direct touch (which is affected by the fat-finger problem), P5 and P9 found that "*[the gestures] work independent from the text size*" due to the indirect placement.

Participants further discussed the perceived effect of text size and explained that the standard text editing mechanisms are only usable for large font sizes while gestures are usable for all sizes ("*I found the standard method indeed not that bad when the text was large. However, for the smaller font, it became obvious that the gesture method is by far the more comfortable and faster method because placing the cursor with the finger [on the front display] is grisly.*" - P1). P7 confirms this and envisioned that "*[he] would sometimes combine it with the standard methods*" - P7). One participant "*[..] did not think that [he] is much better with the gestures than with the normal methods*" (P11).

Impressions on Text Selection Gestures. Ten participants were positive towards the text selection gestures while two provided negative feedback. The majority of the participants found the text selection gestures "*intuitive*" (P12), "*helpful*" (P5, P8), "*very good*" (P6) and "*quick*" (P1, P2, P11). In general, participants "*found it great that a whole word can be selected with a quick gesture*" (P11). Two participants (P3, P13) found the text selection gestures difficult to perform since holding with the thumb while performing input on the rear is uncommon for them.

Impression on the Clipboard Gestures. Eight participants provided positive, two neutral, and two negative comments on the clipboard gestures. In general, participants found the clipboard gesture "very good" (P6), "easy" (P10), "optimal" (P4), and "really fast compared to the normal touch version" (P1, P4, P12). P5 found the gesture fun to perform, "*especially since the cutting gesture feels like a claw.*" While P2 and P3 were neutral towards the clipboard gestures, P2 felt that "*it was not very different to normal touch*" and P3 felt that "*[she] does not need the cut feature that often.*" In terms of negative feedback, P11 found that "*it is sometimes difficult to move two fingers on the back*" while P7 "*finds buttons a little bit easier since [he] can just tap them instead of drawing a gesture.*"

Impression on the Tab Switching Gestures. Two participants were positive towards the tab switching gestures while ten provided negative comments. While P5 and P2 found tab switching "*intuitive*" and "*fine,*" other participants reportedly commented that "*they would rather switch tabs with the [standard method]*" (P3, P4, P8, P5, P7, P12) especially since "*the gestures were difficult to perform*" (P4, P9). P4 and P5 thus suggested to place the tab switching gestures on the "*side close to the volume buttons*" (P4) and "*the top side*" (P5).

Feedback on the Implementation. We asked participants about their impressions after using our implementation of the text-editing gestures. Four participants commended the implementation since they did not encounter any unexpected results which affected their perceived usability (e.g., "*the recognition rate was fine!*" – P3; "*I completed all tasks and it worked fine*" – P4). The eight remaining participants also completed their tasks successfully, but some gestures did not work well for them. For example, P7 found that "*placing the cursor and other [operation such as text selection and clipboard] worked fine, but gestures on the bottom side [for tab switching] did not work well for me.*" This conforms with comments by P1, P2, P3, P5, and P12. P10 further mentioned

a situation in which the cursor unexpectedly moved to the beginning of the line ("*most gestures worked quite well, but I unintentionally jumped to the beginning of the sentence once because I moved my hand*").

7.5 Discussion

We conducted an evaluation in which we focused on three aspects: (i) validating the model accuracy with a new set of participants which were not involved in the data collection; (ii) collecting feedback on the gesture set, including ease and goodness as proposed by Wobbrock et al. [104] as well as qualitative feedback based on an evaluation within a realistic scenario; and (iii) collecting qualitative feedback on the usability of a prototype of the implemented gestures.

Model Validation. The model validation resulted in an accuracy which is close to the test accuracy in the development phase (79.19 % validation vs. 80.92 % test for 12 classes). Thus, we can assume that our model did not overfit to the participants of the data collection study. In general, these results indicate that BoD gestures on FTSPs are feasible with a usable accuracy. While our prototype provided capacitive images with a resolution of $4.1\,mm \times 4.1\,mm$ per pixel which is common for mutual capacitive touchscreens, we expect the accuracy to further improve with a higher resolution based on sensing techniques such as frustrated total internal reflection [26] and infrared sensing integrated into the LCD layer similar to the SUR40 by Samsung.

Feedback on the Gesture Set. The rated goodness and ease of all gestures (except the tab switching gestures) are in the positive range and in line with previous work [5, 92, 104]. This indicates that participants found the gestures easy to perform on our smartphone prototype and that they match their intended purpose.

We further evaluated the gesture set in realistic text-editing scenarios and compared it with Android's standard text-editing mechanisms. Participants rated the perceived easiness, speed, comfort, suitability, and effort after each condition. We adapted these properties from studies presented in previous work [45]. The results revealed that the caret placing, text selection, and clipboard access gestures are generally rated higher than their standard text-editing counterparts. This conforms with the results from the interviews in which the majority of the participants preferred the gestures. Participants further mentioned in the interviews that the usability of the standard text-editing mechanisms decreases with smaller font sizes while the gestures' usability stay consistent for all font sizes. This is also visible in the ratings Figure 11. The tab switching gestures were generally rated as worse than simply touching a tab to switch. In summary, these results show that on-device gestures to activate text-editing operations is feasible with a usable accuracy while perceived as more usable than the standard text editing operations implemented in recent touch-based operating systems.

Feedback on the Prototype Implementation. We asked participants for feedback on our prototypical implementation of the gesture set after they used them in another text-editing scenario. All participants successfully completed the text-editing task using our implementation. While four participants did not activate any function unintentionally, the remaining eight participants encountered unexpected cursor movements due to recognition errors. The interviews and observations of the experimenter revealed that recognition errors are a result of hand grip changes, performing a gesture in a way which does not conform with the data collection study (e.g., different types of executions for the tab switching gestures due to ergonomic constraints), and the similarity of a number of gestures (e.g., swiping down vs. double-tap and then swiping down) due to the low frame rate of the touch sensor. In summary, the feedback showed that our prototype

already enables the use of on-device gestures as shortcuts to text editing operations with a usable accuracy and noticeable improvements over the standard text editing mechanisms.

8 GENERAL DISCUSSION

We presented four studies which focused on bringing shortcuts frequently used on hardware keyboards to FTSP. In the first study, we analyzed shortcuts frequently used on hardware keyboards and interviewed the participants about their opinion on text editing on different devices. Based on the most frequently used shortcuts, we then conducted an elicitation study to derive a user-defined gesture set to provide these shortcuts on FTSPs. In the third and fourth study, we implemented the user-defined gesture set on a fully functional FTSP to evaluate it and gather feedback from potential users. Thereby, we first collected a ground truth dataset of gestures performed on the FTSP and then trained a gesture recognizer. We used our model within representative text-editing tasks to gather qualitative feedback on the gestures, their usability in realistic text-editing scenarios, as well as the recognition accuracy to assess the feasibility of such gestures.

8.1 Use of Shortcuts on Hardware Keyboards and Mobile Devices

The first study explores a set of frequently used gestures on hardware keyboards with a focus on text editing and programming in an in-the-wild setting. Post-study interviews further revealed challenges and common expectations related to the lack of shortcuts on mobile devices. As takeaways, we found that users generally perceive recent methods of providing shortcuts on mobile devices (e.g., long-presses) as inconvenient and laborious. Performing a long-press takes time and interrupts the input flow while nested menus require multiple inputs. Moreover, our results indicate that users perceive placing the caret as challenging especially due to the fat-finger problem. We further identified that users value single-handed interactions especially in mobile scenarios (e.g., while on the go). With around 800 shortcuts performed on hardware keyboards per day, our participants commonly agree that shortcuts on mobile devices are an essential feature which needs to be refined in order to be effective. With improved shortcut mechanisms, our participants envision that mobile devices could become a viable alternative to desktop computers especially for complex tasks in mobile scenarios.

8.2 Gesture Set for Providing Shortcuts on Mobile Devices

The second study resulted in a user-defined gesture set that provides shortcuts to frequently used functions for text editing. An analysis of the gesture characteristics, their taxonomy, as well as the mental model of our participants revealed a wide range of further takeaways that can inform the design of general gestures for FTSPs. Among others, participants proposed gestures based on their previous technical experiences to consider the consistency of gestures between different applications and devices [17] and their memorability [54]. In addition to the consistency between devices, our participants also considered the consistency within the gesture set itself such as by introducing the concept of modifiers (e.g., holding the front screen with the thumb similar to a modifier key). Due to the full-surface touch capability of FTSPs, we further observed that participants preferred compound gestures to avoid unintentional gesture triggers for functions which are difficult to recover.

8.3 Gesture Set Implementation and Evaluation

In the third study, we collected a ground truth dataset for implementing the user-defined gesture set on FTSPs. Due to the low-resolution of the finger imprints, additional unintended finger movements (e.g., to secure a firm grip [50]), and multiple touches (instead of single 2D coordinates as it is the case with gestures on common touchscreens), it is not feasible to use simple heuristics or

algorithms such as $1 [105]. Thus, we used a CNN-LSTM to train a gesture recognizer that enables the use of the gestures with an accuracy of 80 %.

The fourth study evaluated the gesture set and our implementation with potential users. Our participants acknowledged the usefulness of our implementation despite a prototypical accuracy of 80 %. The evaluation based on realistic text-editing scenarios further revealed a number of findings that could be leveraged in future systems to improve text editing. Our BoD gestures were effective regardless of the font size which helps users to keep an overview while editing long texts. As a side effect, we also found that the gestures affect how the phone is held which designers need to keep in mind for developing gestural input on FTSPs.

In summary, we contributed to an understanding of shortcut usage on hardware keyboards, a gesture set to apply these gestures to FTSPs, an implementation thereof, as well as an evaluation which shows that providing frequently used shortcuts improves the usability of mobile text editing. With the recent trend of foldable smartphones and input controls beyond the touchscreen, providing these types of shortcuts could improve a wide range of mobile applications on future smartphones.

9 CONCLUSION

Smartphones have the potential to replace desktop computers for a wide range of tasks. The main tasks performed on computing devices include reading, writing, as well as editing text. While advances in the field of mobile text entry have enabled input speeds that are comparable to hardware keyboards, the lack of shortcuts for common text-editing operations poses a major challenge which affects the usability. The general contribution of this article is an implemented and evaluated gesture set to provide shortcuts to frequently used functions for text editing. This improves the usability of text editing.

In a series of studies, we first gained an overview of shortcuts which were used most frequently by expert users on desktop computers. We studied shortcuts on desktop computers as they are the primary device for text editing. In a second step, we then elicited a user-defined gesture set to provide the most frequently used shortcuts on FTSP. With the recent trend of foldable smartphones and a wide range of previous work on full-touch devices, we envision that future commodity smartphones will offer these touch sensing capabilities. We evaluated the user-defined gesture set by implementing them on an actual FTSP mainly based on deep learning. Even with a rather limited prototype, we found that our gesture set can already be recognized with a usable accuracy of 80 %. Moreover, we found that potential users perceived the use of our gestures as easier, faster, and even more comfortable than the recent methods to access text-editing operations such as cursor placement, text selection, and clipboard management on mobile devices. In summary, results of our evaluation showed that using gestures on FTSPs for text-editing shortcuts is feasible while being perceived as intuitive and faster by users.

There are multiple directions to further increase the recognition accuracy. We used an CNN for feature extraction, an LSTM for processing a time series of feature, and trained both together end-to-end. Future work could decouple the LSTM and the CNN. The feature extraction could be replaced by other deep learning models (e.g., [42, 57, 84]). It could also be replaced by classical implementations, as used by current commercial devices to determine fingers' position on multi-touch touchscreens, potentially combined with algorithms for finger identification [23, 49]. Being able to reliably identify the position of each finger would enable to replace the LSTM by powerful classical gesture recognizers, such as the $N family [2] to improve the recognition accuracy.

While this work is a first milestone towards providing shortcuts for mobile text editing on FTSPs, future work could investigate whether mobile text editing requires further or other types of frequently used actions compared to hardware keyboards. This helps to refine our gesture set

especially with a focus on the attributes of mobile devices such as a rather small display and the fat-finger problem. Moreover, we envision our gesture set to be used and refined for a wide range of applications beyond text editing. Similar to established shortcuts on hardware keyboards which are standardized across different applications (e.g., switching tabs, select all, or copy), future work could apply our user-defined gesture set to other application domains such as web browsing and content management. Furthermore, we evaluated our gesture set on a FTSP prototype within a lab setting. With mass-market smartphones that offer full-touch sensing capabilities (e.g., foldable smartphones) soon being available, future work could evaluate our gesture set in a long-term study in which users use our gestures in a wide range of situations such as while walking or commuting.

AUTHOR STATEMENT

This work is original and has not been previously published in archival form.

REFERENCES

[1] Jason Alexander, Andy Cockburn, and Richard Lobb. 2008. AppMonitor: A tool for recording user actions in unmodified Windows applications. *Behavior Research Methods* 40, 2 (2008), 413–421. DOI: https://doi.org/10.3758/BRM.40.2.413

[2] Lisa Anthony and Jacob O. Wobbrock. 2012. $N-protractor: A fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012 (GI'12)*. Canadian Information Processing Society, 117–120. Retrieved from http://dl.acm.org/citation.cfm?id=2305276.2305296.

[3] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 2017. Structured pruning of deep convolutional neural networks. *Journal on Emerging Technologies in Computing Systems* 13, 3 (Feb. 2017), 18 pages. DOI: https://doi.org/10.1145/3005348

[4] Patrick Bader, Valentin Schwind, Niels Henze, Stefan Schneegass, Nora Broy, and Albrecht Schmidt. 2014. Design and evaluation of a layered handheld 3D display with touch-sensitive front and back. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational (NordiCHI'14)*. ACM, New York, NY, 315–318. DOI: https://doi.org/10.1145/2639189.2639257

[5] Gilles Bailly, Thomas Pietrzak, Jonathan Deber, and Daniel J. Wigdor. 2013. MéTamorphe: Augmenting hotkey usage with actuated keys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 563–572. DOI: https://doi.org/10.1145/2470654.2470734

[6] Olivier Bau and Wendy E. Mackay. 2008. OctoPocus: A dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology (UIST'08)*. ACM, New York, NY, 37–46. DOI: https://doi.org/10.1145/1449715.1449724

[7] Patrick Baudisch and Gerry Chu. 2009. Back-of-device interaction allows creating very small touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. ACM, New York, NY, 1923–1932. DOI: https://doi.org/10.1145/1518701.1518995

[8] Xiaojun Bi and Shumin Zhai. 2016. IJQwerty: What difference does one key change make? Gesture typing keyboard optimization bounded by one key position change from qwerty. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*. ACM, New York, NY, 49–58. DOI: https://doi.org/10.1145/2858036.2858421

[9] John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry* 189, 194 (1996), 4–7.

[10] Wook Chang, Kee Eung Kim, Hyunjeong Lee, Joon Kee Cho, Byung Seok Soh, Jung Hyun Shim, Gyunghye Yang, Sung-jung Cho, and Joonah Park. 2006. Recognition of grip-patterns by using capacitive touch sensors. In *Proceedings of the 2006 IEEE International Symposium on Industrial Electronics*. Vol. 4. 2936–2941. DOI: https://doi.org/10.1109/ISIE.2006.296083

[11] Lung-Pan Cheng, Fang-I. Hsiao, Yen-Ting Liu, and Mike Y. Chen. 2012. iRotate grasp: Automatic screen rotation based on grasp of mobile devices. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST Adjunct Proceedings'12)*. ACM, New York, NY, 15–16. DOI: https://doi.org/10.1145/2380296.2380305

[12] Lung-Pan Cheng, Hsiang-Sheng Liang, Che-Yang Wu, and Mike Y. Chen. 2013. iGrasp: Grasp-based adaptive keyboard for mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 3037–3046. DOI: https://doi.org/10.1145/2470654.2481422

[13] CooTek. 2019. TouchPal. Retrieved April 23, 2020 from http://www.touchpal.com/.

[14] Gennaro Costagliola, Vittorio Fuccella, and Michele Di Capua. 2011. Text entry with KeyScretch. In *Proceedings of the 16th International Conference on Intelligent User Interfaces (IUI'11)*. ACM, New York, NY, 277–286. DOI: https://doi.org/10.1145/1943403.1943445

[15] Wenzhe Cui, Jingjie Zheng, Blaine Lewis, Daniel Vogel, and Xiaojun Bi. 2019. HotStrokes: Word-gesture shortcuts on a trackpad. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI'19)*. ACM, New York, NY, Article 165, 13 pages. DOI: https://doi.org/10.1145/3290605.3300395

[16] Alexander De Luca, Emanuel von Zezschwitz, Ngo Dieu Huong Nguyen, Max-Emanuel Maurer, Elisa Rubegni, Marcello Paolo Scipioni, and Marc Langheinrich. 2013. Back-of-device authentication on smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 2389–2398. DOI: https://doi.org/10.1145/2470654.2481330

[17] Tilman Dingler, Rufat Rzayev, Alireza Sahami Shirazi, and Niels Henze. 2018. Designing consistent gestures across device types: Eliciting RSVP controls for phone, watch, and glasses. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18)*. ACM, New York, NY, Article 419, 12 pages. DOI: https://doi.org/10.1145/3173574.3173993

[18] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. 2017. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 4 (April 2017), 677–691. DOI: https://doi.org/10.1109/TPAMI.2016.2599174

[19] Yin Fan, Xiangju Lu, Dian Li, and Yuanliu Liu. 2016. Video-based emotion recognition using CNN-RNN and C3D hybrid networks. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction (ICMI'16)*. ACM, New York, NY, 445–450. DOI: https://doi.org/10.1145/2993148.2997632

[20] Anna Maria Feit, Daryl Weir, and Antti Oulasvirta. 2016. How we type: Movement strategies and performance in everyday typing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*. ACM, New York, NY, 4262–4273. DOI: https://doi.org/10.1145/2858036.2858233

[21] Dustin Freeman, Hrvoje Benko, Meredith Ringel Morris, and Daniel Wigdor. 2009. ShadowGuides: Visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS'09)*. ACM, New York, NY, 165–172. DOI: https://doi.org/10.1145/1731903.1731935

[22] Vittorio Fuccella, Poika Isokoski, and Benoit Martin. 2013. 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)*. ACM, New York, NY, 2785–2794. DOI: https://doi.org/10.1145/2470654.2481385

[23] Hyunjae Gil, DoYoung Lee, Seunggyu Im, and Ian Oakley. 2017. TriTap: Identifying finger touches on smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17)*. ACM, New York, NY, 3879–3890. DOI: https://doi.org/10.1145/3025453.3025561

[24] Google Commerce Ltd. 2019. Google Docs for Android. Retrieved April 23, 2020 from https://play.google.com/store/apps/details?id=com.google.android.apps.docs.editors.docs.

[25] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. 2007. Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM, New York, NY, 1591–1600. DOI: https://doi.org/10.1145/1240624.1240865

[26] Jefferson Y. Han. 2005. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology (UIST'05)*. ACM, New York, NY, 115–118. DOI: https://doi.org/10.1145/1095034.1095054

[27] Song Han, Huizi Mao, and William J. Dally. 2015. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR* abs/1510.00149 (2015). arxiv:1510.00149 http://arxiv.org/abs/1510.00149.

[28] Gunnar Harboe and Elaine M. Huang. 2015. Real-world affinity diagramming practices: Bridging the paper-digital gap. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, New York, NY, 95–104. DOI: https://doi.org/10.1145/2702123.2702561

[29] Niels Henze, Enrico Rukzio, and Susanne Boll. 2011. 100,000,000 taps: Analysis and improvement of touch performance in the large. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI'11)*. ACM, New York, NY, 133–142. DOI: https://doi.org/10.1145/2037373.2037395

[30] Niels Henze, Enrico Rukzio, and Susanne Boll. 2012. Observational and experimental investigation of typing behaviour using virtual keyboards for mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'12)*. ACM, New York, NY, 2659–2668. DOI: https://doi.org/10.1145/2207676.2208658

[31] David Holman, Andreas Hollatz, Amartya Banerjee, and Roel Vertegaal. 2013. Unifone: Designing for auxiliary finger input in one-handed mobile interactions. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI'13)*. ACM, New York, NY, 177–184. DOI: https://doi.org/10.1145/2460625.2460653

[32] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. 2003. A practical guide to support vector classification. (2003).

[33] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (PCML'15)*. 448–456. http://proceedings.mlr.press/v37/ioffe15.pdf.

[34] Poika Isokoski. 2004. Performance of menu-augmented soft keyboards. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'04)*. ACM, New York, NY, 423–430. DOI: https://doi.org/10.1145/985692.985746

[35] John Karat, James E. McDonald, and Matt Anderson. 1986. A comparison of menu selection techniques: Touch panel, mouse and keyboard. *International Journal of Man-Machine Studies* 25, 1 (1986), 73–88. DOI: https://doi.org/10.1016/S0020-7373(86)80034-7

[36] Kee-Eung Kim, Wook Chang, Sung-Jung Cho, Junghyun Shim, Hyunjeong Lee, Joonah Park, Youngbeom Lee, and Sangryoung Kim. 2006. Hand grip pattern recognition for mobile user interfaces. In *Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence*.

[37] Sunjun Kim, Jihyun Yu, and Geehyuk Lee. 2012. Interaction techniques for unreachable objects on the touchscreen. In *Proceedings of the 24th Australian Computer-Human Interaction Conference (OzCHI'12)*. ACM, New York, NY, 295–298. DOI: https://doi.org/10.1145/2414536.2414585

[38] Anne Köpsel and Nikola Bubalo. 2015. Benefiting from legacy bias. *Interactions* 22, 5 (Aug. 2015), 44–47. DOI: https://doi.org/10.1145/2803169

[39] Per Ola Kristensson and Keith Vertanen. 2012. Performance comparisons of phrase sets and presentation styles for text entry evaluations. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces (IUI'12)*. ACM, New York, NY, 29–32. DOI: https://doi.org/10.1145/2166966.2166972

[40] Per Ola Kristensson and Shumin Zhai. 2004. SHARK2: A large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST'04)*. ACM, New York, NY, 43–52. DOI: https://doi.org/10.1145/1029632.1029640

[41] Per Ola Kristensson and Shumin Zhai. 2007. Command strokes with and without preview: Using pen gestures on keyboard for command selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM, New York, NY, 1137–1146. DOI: https://doi.org/10.1145/1240624.1240797

[42] Abinaya Kumar, Aishwarya Radjesh, Sven Mayer, and Huy Viet Le. 2019. Improving the input accuracy of touchscreens using deep learning. In *Proceedings of the Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA'19)*. ACM, New York, NY, Article Paper LBW1514, 6 pages. DOI: https://doi.org/10.1145/3290607.3312928

[43] David M. Lane, H. Albert Napier, S. Camille Peres, and Aniko Sandor. 2005. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction* 18, 2 (2005), 133–144. DOI: https://doi.org/10.1207/s15327590ijhc1802_1

[44] Huy Viet Le, Patrick Bader, Thomas Kosch, and Niels Henze. 2016. Investigating screen shifting techniques to improve one-handed smartphone usage. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction (NordiCHI'16)*. ACM, New York, NY, Article 27, 10 pages. DOI: https://doi.org/10.1145/2971485.2971562

[45] Huy Viet Le, Thomas Kosch, Patrick Bader, Sven Mayer, and Niels Henze. 2018. PalmTouch: Using the palm as an additional input modality on commodity smartphones. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18)*. ACM, New York, NY, 10. DOI: https://doi.org/10.1145/3173574.3173934

[46] Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. 2017. A smartphone prototype for touch interaction on the whole device surface. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI'17)*. ACM, New York, NY. DOI: https://doi.org/10.1145/3098279.3122143

[47] Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. 2018. Fingers' range and comfortable area for one-handed smartphone interaction beyond the touchscreen. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18)*. ACM, New York, NY, Article Paper 31, 12 pages. DOI: https://doi.org/10.1145/3173574.3173605

[48] Huy Viet Le, Sven Mayer, and Niels Henze. 2018. InfiniTouch: Finger-aware interaction on fully touch sensitive smartphones. In *Proceedings of the 31th Annual ACM Symposium on User Interface Software and Technology (UIST'18)*. ACM, New York, NY, 14. DOI: https://doi.org/10.1145/3242587.3242605

[49] Huy Viet Le, Sven Mayer, and Niels Henze. 2019. Investigating the feasibility of finger identification on capacitive touchscreens using deep learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI'19)*. ACM, New York, NY, 637–649. DOI: https://doi.org/10.1145/3301275.3302295

[50] Huy Viet Le, Sven Mayer, Benedict Steuerlein, and Niels Henze. 2019. Investigating unintended inputs for one-handed touch interaction beyond the touchscreen. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'19)*. ACM, New York, NY, Article Article 34, 14 pages. DOI: https://doi.org/10.1145/3338286.3340145

[51] Huy Viet Le, Sven Mayer, Katrin Wolf, and Niels Henze. 2016. Finger placement and hand grasp during smartphone interaction. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA'16)*. ACM, New York, NY, 2576–2584. DOI: https://doi.org/10.1145/2851581.2892462

[52] Yang Li. 2010. Gesture search: A tool for fast mobile data access. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST'10)*. ACM, New York, NY, 87–96. DOI: https://doi.org/10.1145/1866029.1866044

[53] Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, and Carl Gutwin. 2013. Promoting hotkey use through rehearsal with ExposeHK. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 573–582. DOI: https://doi.org/10.1145/2470654.2470735

[54] Joseph Malloch, Carla F. Griggio, Joanna McGrenere, and Wendy E. Mackay. 2017. Fieldward and pathward: Dynamic guides for defining your own gestures. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17)*. ACM, New York, NY, 4266–4277. DOI: https://doi.org/10.1145/3025453.3025764

[55] Jennifer Mankoff and Gregory D. Abowd. 1998. Cirrin: A word-level unistroke keyboard for pen input. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology (UIST'98)*. ACM, New York, NY, 213–214. DOI: https://doi.org/10.1145/288392.288611

[56] Sven Mayer, Huy Viet Le, Markus Funk, and Niels Henze. 2019. Finding the sweet spot: Analyzing unrestricted touchscreen interaction in-the-wild. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces (ISS'19)*. ACM, New York, NY, 171–179. DOI: https://doi.org/10.1145/3343055.3359705

[57] Sven Mayer, Huy Viet Le, and Niels Henze. 2017. Estimating the finger orientation on capacitive touchscreens using convolutional neural networks. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS'17)*. ACM, New York, NY, 220–229. DOI: https://doi.org/10.1145/3132272.3134130

[58] Hugh McLoone, Ken Hinckley, and Edward Cutrell. 2003. Bimanual interaction on the Microsoft Office keyboard. In *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*. 49–56.

[59] Microsoft Corporation. 2019. Google Play: Microsoft Excel. Retrieved April 23, 2020 from https://play.google.com/store/apps/details?id=com.microsoft.office.excel.

[60] Microsoft Corporation. 2019. Google Play: Microsoft PowerPoint. Retrieved April 23, 2020 from https://play.google.com/store/apps/details?id=com.microsoft.office.powerpoint.

[61] Microsoft Corporation. 2019. Google Play: Microsoft Word. Retrieved April 23, 2020 from https://play.google.com/store/apps/details?id=com.microsoft.office.word.

[62] Takashi Miyaki and Jun Rekimoto. 2009. GraspZoom: Zooming and scrolling control model for single-handed mobile interaction. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'09)*. ACM, New York, NY, Article 11, 4 pages. DOI: https://doi.org/10.1145/1613858.1613872

[63] Mohammad Faizuddin Mohd Noor, Andrew Ramsay, Stephen Hughes, Simon Rogers, John Williamson, and Roderick Murray-Smith. 2014. 28 frames later: Predicting screen touches from back-of-device grip changes. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI'14)*. ACM, New York, NY, 2005–2008. DOI: https://doi.org/10.1145/2556288.2557148

[64] Meredith Ringel Morris, Andreea Danielescu, Steven Drucker, Danyel Fisher, Bongshin Lee, m. c. schraefel, and Jacob O. Wobbrock. 2014. Reducing legacy bias in gesture elicitation studies. *Interactions* 21, 3 (May 2014), 40–45. DOI: https://doi.org/10.1145/2591689

[65] Meredith Ringel Morris, Jacob O. Wobbrock, and Andrew D. Wilson. 2010. Understanding users' preferences for surface gestures. In *Proceedings of Graphics Interface 2010 (GI'10)*. Canadian Information Processing Society, 261–268. Retrieved from http://dl.acm.org/citation.cfm?id=1839214.1839260.

[66] Miguel A. Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. 2013. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 1099–1108. DOI: https://doi.org/10.1145/2470654.2466142

[67] Jakob Nielsen. 1992. Finding usability problems through heuristic evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'92)*. ACM, New York, NY, 373–380. DOI: https://doi.org/10.1145/142750.142834

[68] Michael Nielsen, Moritz Störring, Thomas B. Moeslund, and Erik Granum. 2004. A procedure for developing intuitive and ergonomic gesture interfaces for HCI. In *Gesture-Based Communication in Human-Computer Interaction*. Springer, Berlin, 409–420.

[69] Nuance Communications. 2019. Swype Keyboard. Retrieved April 23, 2020 from http://www.swype.com/.

[70] Daniel L. Odell, Richard C. Davis, Andrew Smith, and Paul K. Wright. 2004. Toolglasses, marking menus, and hotkeys: A comparison of one and two-handed command selection techniques. In *Proceedings of Graphics Interface 2004 (GI'04)*. Canadian Human-Computer Communications Society, 17–24. Retrieved from http://dl.acm.org/citation.cfm?id=1006058.1006061.

[71] Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen, and Per Ola Kristensson. 2013. Improving two-thumb text entry on touchscreen devices. In *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems (CHI'13). ACM, New York, NY, 2765–2774. DOI: https://doi.org/10.1145/2470654.2481383

[72] Benjamin Poppinga, Alireza Sahami Shirazi, Niels Henze, Wilko Heuten, and Susanne Boll. 2014. Understanding shortcut gestures on mobile touch devices. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI'14)*. ACM, New York, NY, 173–182. DOI: https://doi.org/10.1145/2628363.2628378

[73] A Poston. 2000. *Human Engineering Design Data Digest. Washington, DC: Department of Defense Human Factors Engineering Technical Advisory Group*.

[74] Shyam Reyal, Shumin Zhai, and Per Ola Kristensson. 2015. Performance and user experience of touchscreen and gesture keyboards in a lab setting and in the wild. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, New York, NY, 679–688. DOI: https://doi.org/10.1145/2702123.2702597

[75] Jochen Rick. 2010. Performance optimizations of virtual keyboards for stroke-based text entry on a touch-based tabletop. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST'10)*. ACM, New York, NY, 77–86. DOI: https://doi.org/10.1145/1866029.1866043

[76] Anne Roudaut, Stéphane Huot, and Eric Lecolinet. 2008. TapTap and MagStick: Improving one-handed target acquisition on small touch-screens. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI'08)*. ACM, New York, NY, 146–153. DOI: https://doi.org/10.1145/1385569.1385594

[77] Jaime Ruiz, Yang Li, and Edward Lank. 2011. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*. ACM, New York, NY, 197–206. DOI: https://doi.org/10.1145/1978942.1978971

[78] Jaime Ruiz and Daniel Vogel. 2015. Soft-constraints to reduce legacy and performance bias to elicit whole-body gestures with low arm fatigue. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, New York, NY, 3347–3350. DOI: https://doi.org/10.1145/2702123.2702583

[79] Joey Scarr, Andy Cockburn, Carl Gutwin, and Andrea Bunt. 2012. Improving command selection with CommandMaps. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'12)*. ACM, New York, NY, 257–266. DOI: https://doi.org/10.1145/2207676.2207713

[80] Joey Scarr, Andy Cockburn, Carl Gutwin, Andrea Bunt, and Jared E. Cechanowicz. 2014. The usability of CommandMaps in realistic tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'14)*. ACM, New York, NY, 2241–2250. DOI: https://doi.org/10.1145/2556288.2556976

[81] Joey Scarr, Andy Cockburn, Carl Gutwin, and Philip Quinn. 2011. Dips and ceilings: Understanding and supporting transitions to expertise in user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*. ACM, New York, NY, 2741–2750. DOI: https://doi.org/10.1145/1978942.1979348

[82] Jean-Baptiste Scheibel, Cyril Pierson, Benoît Martin, Nathan Godard, Vittorio Fuccella, and Poika Isokoski. 2013. Virtual stick in caret positioning on touch screens. In *Proceedings of the 25th Conference on L'Interaction Homme-Machine (IHM'13)*. ACM, New York, NY, Article 107, 8 pages. DOI: https://doi.org/10.1145/2534903.2534918

[83] Ben Schneiderman. 1986. Eight golden rules of interface design. In *Disponible en*.

[84] Robin Schweigert, Jan Leusmann, Simon Hagenmayer, Maximilian Weiß, Huy Viet Le, Sven Mayer, and Andreas Bulling. 2019. KnuckleTouch: Enabling knuckle gestures on capacitive touchscreens using deep learning. In *Proceedings of Mensch Und Computer 2019 (MuC'19)*. ACM, New York, NY, 387–397. DOI: https://doi.org/10.1145/3340764.3340767

[85] Erh-li Early Shen, Sung-sheng Daniel Tsai, Hao-hua Chu, Yung-jen Jane Hsu, and Chi-wen Euro Chen. 2009. Double-side multi-touch input for mobile devices. In *Proceedings of the CHI'09 Extended Abstracts on Human Factors in Computing Systems (CHI EA'09)*. ACM, New York, NY, 4339–4344. DOI: https://doi.org/10.1145/1520340.1520663

[86] Shaikh Shawon Arefin Shimon, Sarah Morrison-Smith, Noah John, Ghazal Fahimi, and Jaime Ruiz. 2015. Exploring user-defined back-of-device gestures for mobile devices. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'15)*. ACM, New York, NY, 227–232. DOI: https://doi.org/10.1145/2785830.2785890

[87] Katie A. Siek, Yvonne Rogers, and Kay H. Connelly. 2005. Fat finger worries: How older and younger users physically interact with PDAs. In *Proceedings of the 2005 IFIP TC13 International Conference on Human-Computer Interaction (INTERACT'05)*. Springer, Berlin, 267–280. DOI: https://doi.org/10.1007/11555261_24

[88] Smith Aaron. 2015. A "Week in the Life" Analysis of Smartphone Users. Retrieved April 23, 2020 from http://www.pewinternet.org/2015/04/01/chapter-three-a-week-in-the-life-analysis-of-smartphone-users/.

[89] Kenji Suzuki, Kazumasa Okabe, Ryuuki Sakamoto, and Daisuke Sakamoto. 2016. Fix and slide: Caret navigation with movable background. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'16)*. ACM, New York, NY, 478–482. DOI: https://doi.org/10.1145/2935334.2935357

[90] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* 4, 2 (2012), 26–31.

[91] TouchType Ltd. 2019. SwiftKey. Retrieved March 23, 2020 from http://swiftkey.com.

[92] Radu-Daniel Vatavu. 2012. User-defined gestures for free-hand TV control. In *Proceedings of the 10th European Conference on Interactive TV and Video (EuroITV'12)*. ACM, New York, NY, 45–48. DOI: https://doi.org/10.1145/2325616.2325626

[93] Radu-Daniel Vatavu. 2019. The dissimilarity-consensus approach to agreement analysis in gesture elicitation studies. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI'19)*. ACM, New York, NY, Article 224, 13 pages. DOI: https://doi.org/10.1145/3290605.3300454

[94] Radu-Daniel Vatavu and Jacob O. Wobbrock. 2015. Formalizing agreement analysis for elicitation studies: New measures, significance test, and toolkit. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, New York, NY, 1325–1334. DOI: https://doi.org/10.1145/2702123.2702223

[95] Radu-Daniel Vatavu and Jacob O. Wobbrock. 2016. Between-subjects elicitation studies: Formalization and tool support. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*. ACM, New York, NY, 3390–3402. DOI: https://doi.org/10.1145/2858036.2858228

[96] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, New York, NY, 659–668. DOI: https://doi.org/10.1145/2702123.2702135

[97] Daniel Vogel and Patrick Baudisch. 2007. Shift: A technique for operating pen-based interfaces using touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM, New York, NY, 657–666. DOI: https://doi.org/10.1145/1240624.1240727

[98] Panagiotis Vogiatzidakis and Panayiotis Koutsabasis. 2018. Gesture elicitation studies for mid-air interaction: A review. *Multimodal Technologies and Interaction* 2, 4 (2018), 65. DOI: https://doi.org/10.3390/mti2040065

[99] Daryl Weir, Simon Rogers, Roderick Murray-Smith, and Markus Löchtefeld. 2012. A user-specific machine learning approach for improving touch accuracy on mobile devices. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST'12)*. ACM, New York, NY, 465–476. DOI: https://doi.org/10.1145/2380116.2380175

[100] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid touch: A see-through mobile device. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST'07)*. ACM, New York, NY, 269–278. DOI: https://doi.org/10.1145/1294211.1294259

[101] Graham Wilson, Stephen Brewster, and Martin Halvey. 2013. Towards utilising one-handed multi-digit pressure input. In *Proceedings of the CHI'13 Extended Abstracts on Human Factors in Computing Systems (CHI EA'13)*. ACM, New York, NY, 1317–1322. DOI: https://doi.org/10.1145/2468356.2468591

[102] Jacob O. Wobbrock, Htet Htet Aung, Brandon Rothrock, and Brad A. Myers. 2005. Maximizing the guessability of symbolic input. In *CHI'05 Extended Abstracts on Human Factors in Computing Systems (CHI EA'05)*. ACM, New York, NY, 1869–1872. DOI: https://doi.org/10.1145/1056808.1057043

[103] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*. ACM, New York, NY, 143–146. DOI: https://doi.org/10.1145/1978942.1978963

[104] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. ACM, New York, NY, 1083–1092. DOI: https://doi.org/10.1145/1518701.1518866

[105] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST'07)*. ACM, New York, NY, 159–168. DOI: https://doi.org/10.1145/1294211.1294238

[106] Katrin Wolf, Christina Dicke, and Raphael Grasset. 2011. Touching the void: Gestures for auditory interfaces. In *Proceedings of the 5th International Conference on Tangible, Embedded, and Embodied Interaction (TEI'11)*. ACM, New York, NY, 305–308. DOI: https://doi.org/10.1145/1935701.1935772

[107] Katrin Wolf, Marilyn McGee-Lennon, and Stephen Brewster. 2012. A study of on-device gestures. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services Companion (MobileHCI'12)*. ACM, New York, NY, 11–16. DOI: https://doi.org/10.1145/2371664.2371669

[108] Katrin Wolf, Robert Schleicher, and Michael Rohs. 2014. Ergonomic characteristics of gestures for front- and back-of-tablets interaction with grasping hands. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI'14)*. ACM, New York, NY, 453–458. DOI: https://doi.org/10.1145/2628363.2634214

[109] Shi Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems (NIPS)*. 802–810. Retrieved from http://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf.

[110] Shumin Zhai and Per Ola Kristensson. 2003. Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'03)*. ACM, New York, NY, 97–104. DOI: https://doi.org/10.1145/642611.642630

[111] Shumin Zhai, Per Ola Kristensson, Pengjun Gong, Michael Greiner, Shilei Allen Peng, Liang Mico Liu, and Anthony Dunnigan. 2009. Shapewriter on the Iphone: From the laboratory to the real world. In *Proceedings of the CHI'09 Extended Abstracts on Human Factors in Computing Systems (CHI EA'09)*. ACM, New York, NY, 2667–2670. DOI: https://doi.org/10.1145/1520340.1520380

[112] Yu Zhang, William Chan, and Navdeep Jaitly. 2017. Very deep convolutional networks for end-to-end speech recognition. In *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'17)*. 4845–4849. DOI: https://doi.org/10.1109/ICASSP.2017.7953077

[113] Jingjie Zheng, Blaine Lewis, Jeff Avery, and Daniel Vogel. 2018. FingerArc and FingerChord: Supporting novice to expert transitions with guided finger-aware shortcuts. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST'18)*. ACM, New York, NY, 347–363. DOI: https://doi.org/10.1145/3242587.3242589

[114] Jingjie Zheng and Daniel Vogel. 2016. Finger-aware shortcuts. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*. ACM, New York, NY, 4274–4285. DOI: https://doi.org/10.1145/2858036.2858355

Received February 2019; revised February 2020; accepted April 2020