Designing Text Revision Strategies in Advanced User Interfaces

by

Yang LI

Student ID Number: 1218006

A dissertation submitted to the
Engineering Course, Department of Engineering,
Graduate School of Engineering,
Kochi University of Technology,
Kochi, Japan

For the degree of
Doctor of Philosophy

March 2021

ABSTRACT

Designing Text Revision Strategies in Advanced User Interfaces

Text input methods are ubiquitous and vital when users attempt to enter text into digital devices for documentation and communication. Compared with typing on conventional user interfaces based on physical keyboards and mice, typing performance varies when users typing on advanced user interfaces (e.g., smartphones, tablets, virtual reality, and mixed reality) due to their diverse limitations and characteristics. With the popularity of mobile devices (e.g., smartphones and tablets) and commercial Virtual Reality (VR) systems, there exists the need of efficient text input methods that are suitable designed for satisfying typing performance.

Keyboards are dominant and default input tools. On smartphones, keyboards are virtualized and customized (with various layouts) in the limited size on the touchscreen. For VR systems, keyboards are often shown virtually in the intangible space without essential haptic feedback. Although those keyboards could help users to finish the basic text input task, efficiency and satisfaction during the text input, however, cannot be guaranteed. The main reason is that the keyboard layout, definition of keys, and related operation patterns are inherited from the physical QWERTY keyboard, which already existed for decades with few adjustments based on various platforms and scenarios.

Generally, text input includes two main parts. The first part is text entry, which mainly focuses on generating text quickly into the computing system based on users' input. The second part is text revision, which is responsible for ensuring that the input text is both typo-free and proper to express the intended meaning of the typist. As far as we know (from the literature), most current text input methods and research focus on enhancing users' typing performance (e.g., faster typing speed). However, little attention was paid to improve text revision efficiency, which is essential to ensure the expression accuracy of the input text. It should be noted, again, that the main target of text input is for documentation (e.g., taking notes) and communication (e.g., discussion with friends via instant messaging or emails) rather than just entering some characters and show them on the screen.

Current text input methods can handle typos and grammar issues easily with auto-correction techniques. Whereas, for more general revision conditions (e.g., revising the word with right spelling but with improper meaning or adding missing words into the sentence), those methods cannot achieve the satisfying text revision experience, especially for smartphones and VR systems.

This dissertation mainly focuses on facilitating text revision on mobile devices and in VR applications. A systematic review was conducted first to summarize the text revision attempts applied in current smartphone and VR text input methods. With the review, we further analyze their commonalities and flaws. It revealed that, for typing with virtual keyboards, most text revision attempts still followed the operation process using the backspace and cursor control, which already existed for decades.

To improve the text revision efficiency on mobile devices, we revisited the existing text revision process and proposed Swap, a replacement-based text revision paradigm, to enhance the text revision performance by minimizing the use of backspace and cursor control. In detail, Swap regards all characters and words as replaceable and independent units. When observing the revision target, users can enter the revised content first and then use it to replace the target. To change the processes for various revision conditions (e.g., inserting, substituting, or deleting a word) into the unified replacement operation, Swap also visualizes some specific functions (e.g., deletion) and allows them to appear in the input string just as regular characters.

Based on the paradigm, we implemented a text revision technique (named Swap) and evaluated its feasibility on smartphones via conducting a comparative user study. Results showed that, compared with the repetitive backspace pressing and imprecise cursor control, Swap simplifies the steps and the number of the potential mode switch process (i.e., from regular typing to revision or from revision to regular typing). Moreover, Swap enables users to keep their regular typing speed during the revision process on smartphones.

In the context of VR environments, a series of techniques have been designed and proposed to enhance the text input performance. However, few researchers put their eyes on the enhancement of text revision. To deeply understand the research status of the current VR text entry, we first did a systematic review and revealed that there lacks the essential consideration on the problem of text revision. Even worse, most current proposed techniques did not include the tools and solutions to handle the need for text revision in VR. Then, we proposed a design space based on caret and backspace to explore the design solutions for enhancing text revision performance in VR applications. With the design space, we further implemented four text revision techniques and evaluated them using a comparative user study. Outcomes of the design space and proposed techniques not only provided a fundamental understanding of VR text revision solutions (with the backspace and caret) but also a comparable basis for evaluating future VR text revision techniques.

During the review of current VR text entry research, we also found that, during text entry, although characters are selected in sequence, there lacks smooth transition among every two selections, which (to some extent) influence the typing speed. Therefore, apart from text revision, an additional study was conducted to enhance the text input efficiency by proposing SewTyping, a novel technique that fully leverages the penetrable feature of the intangible interface and the daily-life sewing metaphor to achieve the fluid and successive text entry behavior just like sewing with a needle on the fabric. We got inspiring results that SewTyping not only improves the typing speed in VR applications but also changes the VR text input as engaging gameplay.

This dissertation shows contributions as follows:

First, the literature review reveals the insufficient attention of current text revision research when designing useful and efficient text input methods. For researchers who are interested in this field, this dissertation can also serve as a systematic overview of text revision.

Second, instead of the conventional text revision paradigm (based on backspace and cursor control), Swap shows an overturning perspective for designers to consider and design text revision techniques on mobile devices and in VR applications at both the process level and practice level.

Third, for VR text input techniques, we point out the lacking of essential considerations of text revision in the previous existing research. Our proposed design space and text revision techniques can both provide a perspective to face this practical problem (text revision in VR) and available options that attempted to enhance users' text revision performance when entering text with virtual keyboards in VR.

Fourth, SewTyping provides a novel way to consider the penetrable feature of the intangible interface and convert it as an advantage when designing the fluid text input operation. Additionally, SewTyping also proposes a new interaction approach (sewing interaction), which sheds light on the novel interface design for VR applications.

Overall, methodologies and results reported in this dissertation will be beneficial for both researchers and practitioners when exploring and implementing text revision techniques to achieve a more satisfying typing performance on mobile devices and in VR applications.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

CHAPTER 1

INTRODUCTION

1.1. Background

Since 1955, physical keyboards appeared officially with computers and allowed users to enter text with them. Till now, although various fast input methods such as voice input and pen-based input appear and provide users with various choices, typing with keyboards is still the mainstream and vital part when users interacting with computers in different forms. Nowadays, the increasing popularity of mobile devices (e.g., smartphones and tablets) and consumer-affordable VR devices not only brings requirements of the more satisfying interaction experience from users but also raises tons of challenges that designers need to solve.

Compared with typing with physical keyboards and mice, typing with smartphones and VR systems is error-prone and time-consuming due to the lack of haptic feedback, limited target size, and/or users' poor capacities when interacting with intangible interfaces. In this situation, the existed interaction design (inherited from physical keyboards) can only feed the basic need of text input without the requirement of efficiency and satisfaction. Moreover, through the literature review and the investigation of current commercial text input solutions, we found that most of the attempts try to enhance the typing speed and accuracy. Here the accuracy mainly focuses on errors that refer to typos and grammar issues. With smart-aid techniques based on Artificial Intelligence (AI) and other algorithms, it is easy to solve the errors mentioned above. However, when all typos and grammar issues are corrected, there still exists the need of revising text content because that the input text cannot express the proper meaning from the typist. For instance, when composing a business email on smartphones or in a VR office software, one can ensure that all the text typo-free and grammar correct, but s/he found that s/he uses a word with a reversed meaning or makes an ambiguous statement. In this situation, based on the current tools (mainly with backspace and cursor control), revising the text will be unnecessarily complex, error-prone, and time-consuming when navigating the cursor and pressing the backspace key.

One may argue that "AI can solve them all". However, there exist two problems hard to avoid. First, with the technique so far, AI cannot accurately understand what users mean when they enter the text. Thus, AI may give some wrong operations, which cause more efforts to recover from it (e.g., auto-completion). Second, when AI and algorithms finish the error correction and regard all text "ok" according to their perspectives, they may not provide help when users revise the text from the context level. In that case, there will be no more choices for users except backspace and cursor.

1.2. Motivation and Objectives

There are four points that motivate the studies in this dissertation. First, as far as we know from the literature, most researchers only focus on chasing for faster typing speed with less attention on text revision. Second, although some researchers attempt to improve the text error correction (typos and grammar issues correction) efficiency, most of them conduct their research without consideration of more general revision conditions. Third, there lacks the attempt to challenge or clarify the rationale and responsibilities of backspace and cursor in error correction and text revision in mobile devices and VR systems. Last but not least, with the increasing trend of VR systems, there exists a strong need to enhance users' text input performance (both text entry and text revision) as the basement to build more effective and engaging experience in VR.

Thus, the objects of this dissertation are:

- To provide a thorough overview about the state-of-art of current text revision research (in both mobile devices and VR systems)
- To propose a new text revision process and tools that can both simplify the interaction process and avoids the potential errors during the revision
- To clarify and extend the design space of VR text revision
- To propose an effective and fluid text entry technique to enhance the typing speed in VR applications

1.3. Dissertation Overview

The details of this dissertation are organized as follows. Chapter 2 shows the holistic overview about the development of keyboards, mice, text error and error types, and error correction vs. text revision. Chapter 3 illustrates the challenges we found about text revision and proposes the new text revision paradigm (named Swap), related techniques, and validation process (through iterative design and a series of user studies). Chapter 4 focuses on investigating text revision in VR systems. The main work is to investigate the design space of VR text revision and proposed a series of techniques to improve user’s text revision performance in VR applications. Chapter 5 focuses on facilitating text entry efficiency and fluidity in VR applications by proposing SewTyping, a novel technique that combines the daily-life sewing metaphor with the penetrable feature of intangible displays. Chapter 6 illustrates the in-depth discussions based on the finding of Chapter 3, 4, and 5. Chapter 7 lists the limitation of the current studies and the future work based on current research conditions. Finally, chapter 8 summarizes the work and highlights the contributions of this dissertation.

CHAPTER 2

LITERATURE REVIEW

This chapter first describes the history and the development of keyboards and text revision tools. Then, this chapter illustrates errors and error correction tools used during text input. Finally, we describe the differences between error correction and text revision and current solutions for text revision.

2.1. Evolution of Keyboard

People started to use machines to impress characters on paper since 1575 with scrittura tattile, the machine invented by Francesco Rampazzetto (*Keyboard - History of the Modern Computer Keyboard*, 2019). With the existence of the first commercially successful typewriter in 1873 (Naskar, 2019), the keyboard with a QWERTY layout became mainstream and allowed users to transcribe text with it. On the keyboard, functions (e.g., shift and backspace), punctuations, and characters are designed as individual buttons and arranged in a rectangular layout.

With the existence of personal computers, keyboards also serve as default input devices to receive users' commands and transcription. The design (e.g., the definition of keys, and the way to transcribe and edit characters) of computer keyboards was inherited from the keyboards on typewriters. Interestingly, computer keyboards also inherited the QWERTY layout, which is previously designed to address the mechanical problems on typewriters. Based on this layout, computer keyboards also include arrow keys to control the caret among input text.

In the 1990s, feature phones became popular and wide-spread to users. Compared with the physical keyboard used in desktops and laptops, the interactive area of feature phones is not sufficient. Thus, designers first integrated characters and numbers into a 12-key physical keyboard and put arrow keys and backspace into a separate area (see Figure 2.1).

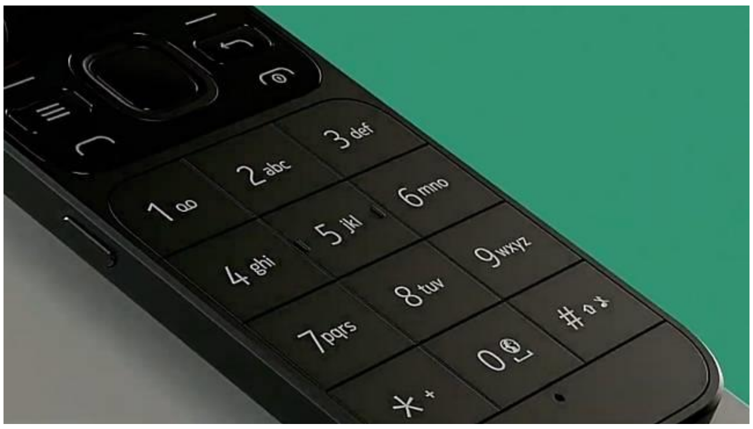


Figure 2.1 A feature phone with a 12-key keyboard. Arrow keys and the backspace key are put above the keyboard.

There were also designers who attempted to shrink the size of the physical QWERTY keyboard and transplant it to feature phones (see Figure 2.2) with essential adjustments. Although this keyboard kept the QWERTY layout, it limited the size of buttons and thus influence the typing speed (Arif, 2015).

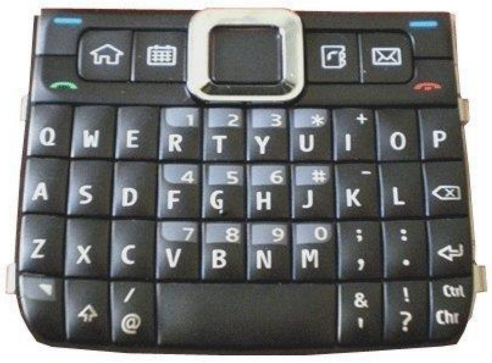


Figure 2.2 A shrinked-size QWERTY keyboard on the feature phone.

With the development of modern mobile operating systems (Android and iOS) and the popularity of touch screens, less space was left for keyboards in a physical form on smartphones and devices with larger screens (e.g., Microsoft Surface and iPad). Thus, virtual keyboards became dominant tools (as an alternative of physical keyboards) for text input. It is feasible to carry a physical keyboard to support users' text input. However, it requires users to bring additional devices (with the loss of mobility and the increased carry weight) with them. To avoid the extra workload of learning to type with a new keyboard, virtual keyboards inherited a similar design of physical keyboards with essential adaptive simplification such as cutting down arrow keys and using the multi-layer design with mode-switch mechanisms.

With the similar layout, users can easily learn from their typing experience with physical keyboards (including the memory of key positions) and adapt it to new devices easily. However, there exist challenges when users typing with the virtual keyboard on touch screens. First, the limited size of keys and the finger occlusion make it difficult to press keys precisely. Second, compared with the mouse (with the scroll wheel and two physical buttons), controlling the caret with finger touch cannot provide satisfying accuracy and various functions directly. Third, although users can call out arrow keys to simulate similar functions, it requires extra time and steps to switch the layout between the QWERTY keyboard and arrow keys (see Figure 2.3).

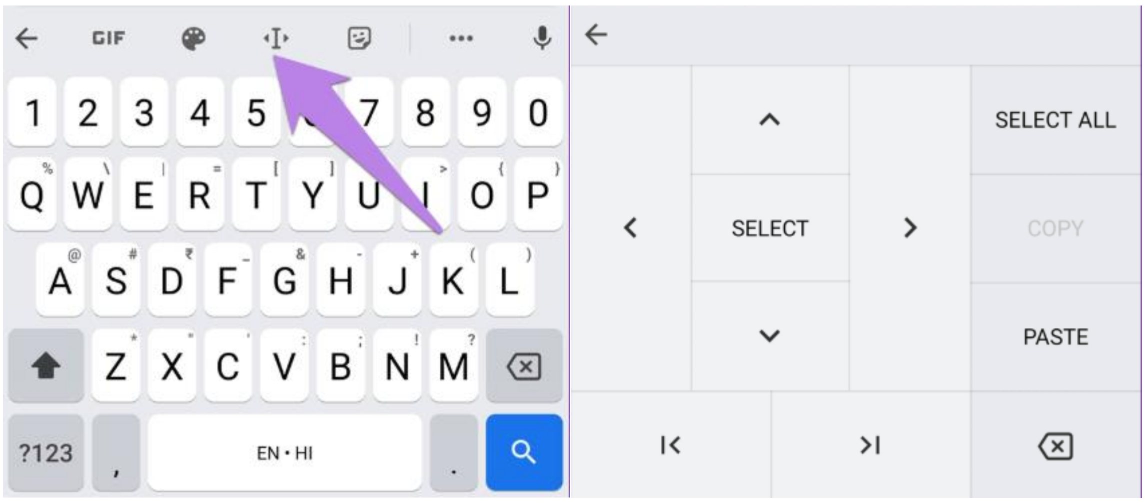


Figure 2.3 Mode switching between QWERTY keyboard and arrow keys on a smartphone virtual keyboard.

In VR applications, the situation got worse than that on mobile devices when typing with virtual keyboards. A review about VR text input (Dube & Arif, 2019) revealed that typing with virtual keyboards cannot achieve satisfying typing performance as that with physical keyboards. There are two reasons for that. First, users need to use bare hands or handheld controllers to contact with keys without haptic feedback (the force resistance from the physical surface when pressing down keys) and few visual and audio feedback for confirming the key pressing during the text input, which both influence the typing fluidity and increase the possibility of mis-operation and typos. Second, for interaction with intangible interfaces, users have the poor capability of sensing the depth information

(Chan et al., 2010) of virtual objects, which leads to uncertainty, mis-operation during the interaction, and relatively low typing speed.

2.2. Error and Error Correction

It is inevitable to commit errors when users interacting with computers. In the context of text input. It is common to observe that users make typos when composing text with keyboards. According to the research (Gentner et al., 1983), all typos can be summarized as three main types: insertion, substitution, and omission. Insertion is the error where include one more letter in the input string (e.g., "helllo" to "hello"). Substitution is the error where one wrong character was entered to the position that the right character should be (e.g., "hrllo" to "hello"). Omission is the error where one character is missing ("helo" to "hello"). For errors with multiple characters redundant, mis-input, or missing, they can also be deconstructed to the three error types mentioned above.

To correct errors during typing, error correction tools and mechanisms are essential and vital parts. Date back to the time of typewriter, erasers and ribbon are the main tools to correct typos with the assistance of backspace key, forward key, and the scroll roll (to move up or down the line). When the typist found the typo, they need to press the backspace key to make the typing position go back to the position where the mistaken character was. Then s/he uses the eraser to clean or the ribbon to cover the mistaken character. After the correction, the typist cannot continue typing until the typing position was moved to the end of the input string.

On PC physical keyboards, the function of the backspace key was updated. When pressing the backspace key, two functions will be conducted at the same time: the caret will go backward with the distance of one letter with the letter after the caret is deleted. With arrow keys and mouse, users can navigate caret to the typo position, finish the correction, and move the caret back for further input. Furthermore, with a mouse, a series of characters can be selected by pressing and holding the button on the mouse, which increases the efficiency of error correction.

For error correction with virtual keyboards on mobile devices and VR systems, it shares a similar operation as that with PC physical keyboards. However, users need to face a series of usability issues to correct typos (as mentioned in Chapter 2.2).

Fortunately, based on the powerful corpus (Grammarly, 2019), algorithms (Levenshtein, 1966), and artificial intelligence methods, typos and grammar issues can be easily detected during the text input process and corrected automatically (e.g., auto-completion) or semi-automatically (e.g., technique labels the typos, users choose the item in a prediction list to correct). With the techniques above, most typos and grammar issues could be addressed with limited attempts from users.

2.3. Text Revision and Text Revision Tools

Apart from correcting typos and grammar issues, there also exist conditions that need typists to revise. For instance, when composing an email on a smartphone, the user can make sure all text is typo-free and grammar correct with auto-correction techniques. However, s/he found that there are some words and sentences that cannot express the intended meaning. In that case, the user must revise them to make sure its expression accuracy.

To finish the revision, backspace and caret are the only tools available. Usually, backspace is effective to finish the correction when the typo is just 3-5 characters away from the caret (Komninos et al., 2018). Beyond that, backspace would lose its effectiveness. Due to the difficulty of controlling the caret and the function limitation of backspace (backspace can only delete one character per pressing), text revision becomes time-consuming and error-prone on mobile devices and VR systems. Even worse, the repetitive use of backspace and imprecise control of caret may lead to cascade mistakes (e.g., multiple attempts to locate the caret) and unexpected time consumption (e.g., deleting the content unnecessarily with backspace).

CHAPTER 3

SWAP-BASED TEXT REVISION TECHNIQUE ON MOBILE DEVICES

This chapter illustrates the design and validation of the novel text revision technique on mobile devices. In detail, we present *Swap*, a novel replacement-based technique to facilitate text revision on mobile devices. We conducted two user studies to validate the feasibility and the effectiveness of Swap compared to traditional text revision techniques. Results showed that Swap reduced efforts in caret control and repetitive backspace pressing during the text revision process. Most participants preferred to use the replacement-based technique rather than backspace and caret. They also commented that the new technique is easy to learn, and it makes text revision rapid and intuitive.

3.1 Introduction

Text revision is a ubiquitous and vital process for text-related tasks such as email composition and instant messaging on mobile devices. Typo-free contents contribute in both formal and informal scenarios to help the reader convey information accurately. Currently, virtual keyboard interfaces offer backspace and caret control tools for erasing characters and navigating the caret. With the help of automatic techniques (e.g., spell checker (Alharbi et al., 2019; Android, 2017) and input guidance (Hagiya et al., 2016)) and writing assistant software (e.g., Grammarly (Grammarly, 2019)), users can also avoid, detect, and/or correct typos and grammar mistakes.

Despite many existing techniques, users still face significant usability issues when revising text on mobile devices. First, navigating the caret with the finger is error-prone and time-consuming, due to finger occlusion (Siek et al., 2005) and small target size (Brewster, 2002; Colle & Hiszem, 2004). Second, auto-correction sometimes introduces confusing, embarrassing, and hard-to-observe errors (Alharbi et al., 2019; Arbesman, 2017; Arif & Stuerzlinger, 2013; Microsoft, 2019), that diminish the benefits of those techniques and introduce extra revision workload to users (Buschek et al., 2018). Third, the character-level backspace makes the revision process time-consuming and error-prone when the revision contains multiple characters and/or happens in the middle of an input string (Arif & Stuerzlinger, 2010). Except for the backspace and caret, there lack efficient tools (and methods) when revising the sentence itself (e.g., modifying the meaning of the sentence by inserting, deleting, or substituting word(s)).

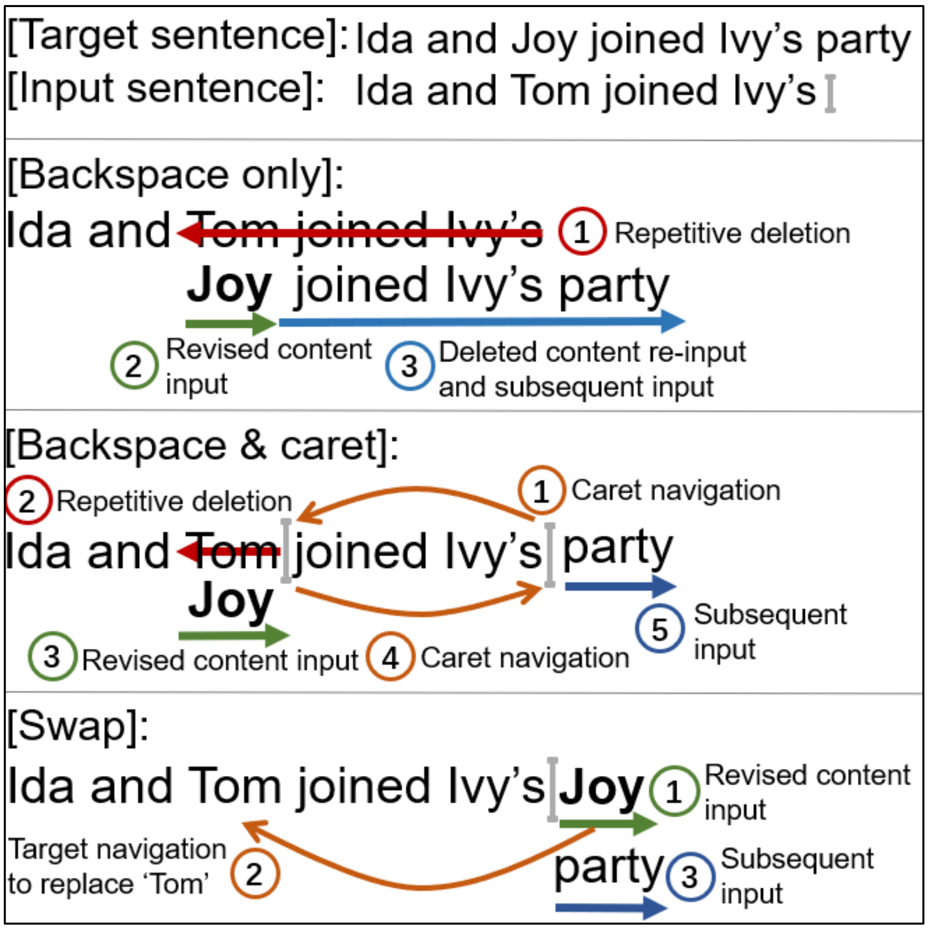

[Target sentence]: Ida and Joy joined Ivy's party
[Input sentence]: Ida and Tom joined Ivy's
[Backspace only]:
Ida and ~~Tom joined Ivy's~~
1 Repetitive deletion
Joy joined Ivy's party
2 Revised content input
3 Deleted content re-input and subsequent input
[Backspace & caret]:
2 Repetitive deletion
1 Caret navigation
Ida and ~~Tom~~ joined Ivy's party
Joy
3 Revised content input
4 Caret navigation
5 Subsequent input
[Swap]:
Ida and Tom joined Ivy's Joy
1 Revised content input
Target navigation to replace 'Tom'
2
party
3 Subsequent input

Figure 3.1 Operation sequences (shown with enclosed numbers) for different techniques when revising the sentence. Swap minimizes the use of backspace and caret by first entering the revised content and then using it to replace the target.

We present Swap, a novel technique that is designed to facilitate text revision on mobile devices by using a unified replacement-based process to address various text revision tasks (e.g., substituting/deleting/inserting a word). Instead of using backspace and caret repetitively, Swap allows users to do the revision by first entering the correct content, and then using it to replace the target (see Figure 3.1). Swap 1) reduces the time consumption and the workload when using the backspace, and 2) avoids moving the caret in order to ensure the quick recovery from revision to the regular text entry process. With Swap, users can focus on the revision itself rather than spending time on auxiliary steps.

This part of the work is organized as follows. First, we conducted a workshop to investigate the pain points of the current text revision interaction on mobile devices. After that, we illustrate the design of Swap. Then, we make the first implementation (Dot Swap) to validate the feasibility of Swap with a user study. Based on that study, we further improve the design, propose an iterative implementation (improved Swap), and evaluated its effectiveness through a user study compared with the conventional backspace and caret. Finally, we discuss Swap technique insights and future work.

3.2 Related Work

In this section, we first review previous research on text error correction, including the tool (backspace and caret) improvement, and current solutions (techniques, algorithms, and commercial products) for enhancing users' error correction performance. Then, we make a summary based on the review to reveal the gap between error correction and text revision.

3.2.1 Text Revision Procedure on Mobile Devices

Generally, the design of text revision on current mobile devices obeys the Object-Action model (Card et al., 1983; Norman, 1995): First, users need to navigate the caret to the target position, and then finish the revision with repetitive backspace pressings (and extra caret control). As text revision often happens during a text entry process, users usually have to relocate the caret for other operations (another text revision or the following text entry). As a result, we summarize the current text revision procedure as "navigate the caret, revise the target, and navigate the caret".

3.2.2 *Improvement of the Backspace*

The conventional backspace serves users with the character-level function of both deleting a character and moving the caret backward (Wikipedia, 2019). Researchers attempt to use hotkeys (Raymond, 2007) and touch pressure (Ali, 2017) to extend deletion granularity of the backspace and decrease repetitive use of the backspace.

Gestures are often used to facilitate error correction in both research (Fuccella et al., 2013) and the industry field (Fleksy, 2019). Smart-restorable backspace (Arif et al., 2016) allows users to quickly delete and store the text after the typo position by swiping left on the backspace key. A similar solution also occurs in (Alharbi et al., 2019). Instead of only having one backspace key on the keyboard, Arif et al. (Arif et al., 2014) removed the backspace key and integrate its function into the whole keyboard. Users can swipe left on any key on the keyboard to trigger the deletion.

3.2.3 *Caret Control Enhancement*

Caret control on mobile devices has been well investigated by researchers. Caret is an indicator for both mobile devices and users to remind them where interactions may happen. The trackpad mode on a virtual keyboard (idownloadblog, 2018), magnifier (Google, 2018a) widget, and selection handle (Google, 2018b) can help users navigate the caret with flexibility and precision. iOS 13 (Apple, 2019) uses the mouse to navigate the caret on tablets. However, those tools introduce extra interaction steps, cognitive load, and the possibility of increased errors.

Users can navigate the caret in both direct and indirect ways. Widgets such as virtual sticks (Scheibel et al., 2013) and arrow keys (Google, 2019; Microsoft, 2019) are integrated into virtual keyboards to simulate the caret control on physical keyboards. By holding one key on the keyboard, Ando et al. leveraged the device tilt (Ando et al., 2018) and slide gesture (Ando et al., 2019) to complete the caret control, target selection, and a particular command (e.g., copy) at the same time. Eady et al. (Eady & Girouard, 2015) built a deformable interface and proposed a bend gesture on the corner of the device to control the caret. Sindhwani et al. (Sindhwani et al., 2019) used a matching algorithm to highlight potential correction position(s) based on users' input. Then, eye movement was used to identify the intended correction position. Instead of moving the caret, Suzuki et al. (Suzuki et al., 2015) attempted to move the caret by moving the background text.

3.2.4 Error Correction

Algorithms and AI-based methods (e.g., natural language processing) are widely used in current typing applications: 1) to detect typos and grammar mistakes and 2) to prevent those issues from appearing in the final text (Arnold et al., 2016). Arif et al. (Arif et al., 2016) used Levenshtein distance (Levenshtein, 1966) to calculate the nearest position for their smart error correction technique. Retype (Sindhwani et al., 2019) used a simple string matching method to filter the potential correction positions by entering a few characters. Variants of visual feedback were used to inform users of detected problems. Maxie Keyboard (Komninos et al., 2015) and Wisetype (Alharbi et al., 2019) used colored shades to highlight typos. Grammarly (Grammarly, 2019) used underlines to remind users of detected errors. Users can do a quick correction by tapping on the error and select a required option provided by the system. Additionally, applications such as Microsoft Word and Gmail also provide auto-correction functions. These systems automatically correct typos as the users tap the space key after entering a word.

3.2.5 Summary

To date, extensive research mentioned above were focused on addressing typos and grammar mistakes. Most of those errors can be solved (semi-)automatically with corpora, machine learning, and auto-correction techniques. However, it may be noted that typos and grammar mistakes are not the only targets for text revision. Except for backspace and caret, there lack efficient tools to deal with conditions such as revising words with unintended meaning, or changing unintended and/or confusing words inserted by incorrect auto-correction (Arif et al., 2016). Furthermore, through relevant literature, we found that most research was focused on only one part (e.g., caret control) of the text revision procedure rather than questioning the text revision procedure itself as well as the rationality of the current procedure and tools used in mobile devices.

3.3 Pre-Study: How Do Users Do Text Revision

Before designing the new text revision technique, we conducted a workshop to investigate strategies of using the backspace and caret in daily text revision scenarios and to explore the challenges when revising text content on mobile devices. The workshop involved 20 participants (age ranged from 24 to 32, 10 females), all of whom have used smartphones for over six years. All participants used instant messaging applications (e.g., Line and WhatsApp) every day. Eight of them compose emails daily on their smartphones. We designed several text revision scenarios (e.g., changing a word in the middle of a sentence when composing an email) and asked them to simulate them on their smartphones. After that, we organized a free discussion with participants to collect their feedback on the use of backspace and caret with a view of improving our design.

Most (18 of 20) participants preferred to use two thumbs for typing on the virtual keyboard. All participants used the backspace (only) for quick revisions if there were only a few characters away from the caret. When editing was required in the middle of the sentence, three participants tended to use backspace only, while the others navigated the caret, executed the correction, and then navigated the caret again to other positions for further actions. When navigating the caret, ten participants used assistive techniques (e.g., selection handles and the magnifier). All participants reported that it is difficult to navigate the caret with their fingers due to the occlusion.

In the free discussion session, most participants regarded the text revision process as "*using the proper content to* ***replace*** *the improper one*". They commented that it was simple but cumbersome to perform that on mobile devices. They felt it "simple" because they had used backspace and caret for years. Participants subconsciously turned the intention to revise into a sequence of steps. They felt the process "cumbersome" because of the limitations of backspace and caret. Participants must spend time on extra but essential steps (e.g., caret control and consecutive deletion) to revise the text. All participants commented that frequent caret control interrupted their typing flow. They must stop typing, move their fingers out of the virtual keyboard to navigate the caret, finish the correction, and restart typing after that (with navigating the caret once more). This situation became worse if participants have multiple words to correct.

3.4 The Design of Swap

Based on the pre-study, we regarded "replacement" as the core design principle to simplify the mobile text revision process. There are two ways to implement the replacement process. One is "type and then select", while the other is "select and then type". We chose the first one to enable participants to enter revision content quickly without the obvious interruption caused by the selection process.

"Replacement" can happen not only between words (e.g., replace "*John*" with "*Sam*" in "*John will come and visit Bob*") but also between functions and words (e.g., deleting "*almost*" by using a symbol (representing the delete function) to replace "*almost*" in "*I am almost ready to go there*").

In the current situation, it is difficult to realize "replacement" in an intuitive way. There are two reasons: 1) with current backspace and the caret control method, there lacks a unified procedure to deal with different revision conditions (e.g., inserting/deleting/substituting a word); 2) the current backspace function cannot appear on the touch screen as a regular character. Thus, we design the replacement-based text revision process and symbolize the backspace as a character to make it visible on the screen.

3.4.1 Interaction Logic

Swap changes the revision into two steps: *Content Preparation* and *Replacement Execution*. With them, Swap unifies various revision procedures into one: enter the content and then replace the target with it.

Content Preparation: The first step of revision is to enter the content for the following replacement. The content includes characters and the symbolized backspace. For instance, if participants want to: 1) insert "*mobile*" between "*in*" and "*text*" or 2) change "*tools*" to "*methods*" in "*... text revision tools show vital importance in text entry tasks.*", they can enter the text ("mobile" and "methods") at the end of the sentence. If the goal is to remove the first "text", participants can enter a symbolized backspace (this will be elaborated later in the following part) representing the command of deleting "text".

Replacement Execution: After the content preparation, the following step is to use the entered content to finish the revision. Specifically, participants can navigate the content and finish the replacement by tapping the target. After the replacement, the caret remains at the end of the input string and ready for further input actions. When revising "*... text edition tools show importance in mobile text entry tasks...*", participants can: 1) replace "*edition*" with "*revision*"; 2) replace the space between "*show*" and "*importance*" with "*vital*"; and 3) use a symbolized backspace to replace "*mobile*".

3.4.2 Symbolized Backspace

Compared with the conventional backspace, we visualize the function of deletion on the screen with the symbolized backspace. The symbolized backspace represents the function of "deleting multiple characters". When participants want to delete one word, they can enter a symbolized backspace, and then use it to replace the word. It may be noted that the symbolized backspace will disappear after the replacement and it will not appear in the final text.

3.5 Dot Swap

Two issues are needed to solve before implementation. First, we need a suitable symbol to represent the function of "deleting multiple characters". Second, we need an approach that helps identify different revision intentions (e.g., inserting or deleting a word) during the revision process. It should also be noted that, when entering the symbol, it should not bring much extra cost to participants (e.g., mode switch, visual search). We addressed those issues by redesigning the use of the dot (".") because it is the common symbol directly shown on the default virtual keyboard. To 1) validate the feasibility of the new text revision process and the symbolized backspace and 2) improve the design of Swap, we implemented Dot Swap and evaluated it with a user study.

Dot Swap used a dot as both the symbolized backspace and the identifier. As shown in Figure 3.2, when participants observed the target for revision, they would first enter the content at the end of the sentence. If participants entered only one dot, this dot would be regarded as the symbolized backspace. If the entered content didn't contain any dot at the beginning, it can be used to substitute the target word. Content marked with two dots as a prefix meant that the content will be inserted into the sentence. After entering, participants tapped the intended target (when inserting a word, Dot Swap calculated the nearest space to the touch point) to finish the revision.

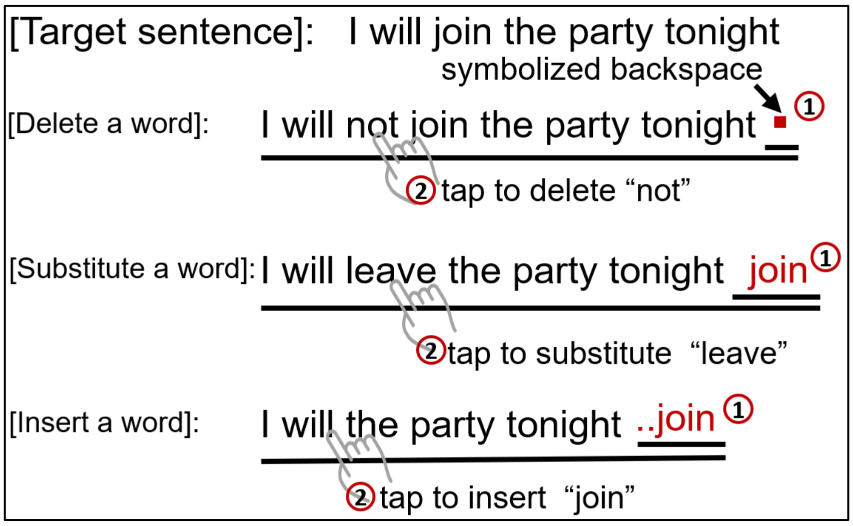

[Target sentence]: I will join the party tonight
symbolized backspace
[Delete a word]: I will not join the party tonight .
①
② tap to delete "not"
[Substitute a word]: I will leave the party tonight join
①
② tap to substitute "leave"
[Insert a word]: I will the party tonight ..join
①
② tap to insert "join"

Figure 3.2 The use of Dot Swap under three revision conditions. Participants first enter the content at the end of the sentence, then use that content to revise the target.

3.6 Study 1 – Feasibility Evaluation of Dot Swap

We conducted a user study to 1) examine the feasibility of the replacement-based process for text revision and 2) compare the text revision efficiency of Dot Swap with the conventional backspace & caret and magnifier.

3.6.1 Apparatus

A custom application was designed to implement the text revision techniques with Android P (SDK version 28) in Java on a Huawei P20 smartphone (5.8 inches, 2244 × 1080 pixel, 149.1 × 70.8 × 7.65mm). The smartphone also recorded participants' key pressing and caret control events during the experiment. We disabled assistive functions such as auto-correction and auto-completion.

3.6.2 Participants

We recruited nine participants (average age 24.67, *SE* = 1.15, three females) for this study. One participant is left-handed. All participants had experience over three years of using mobile devices and preferred to use two thumbs when typing on the touch screen.

3.6.3 Procedure and Task

The study used a within-subjects design to compare text revision performance of conventional backspace & caret (as the baseline), magnifier (Google, 2018a) (when participants move the caret, a virtual lens appears and follows the finger position with the content below the fingertip), and Dot Swap. We used the memorable test set from the Enron Mobile Email Dataset (Vertanen & Kristensson, 2011).

We set two phases for each trial to simulate the revision behavior during the text entry process (e.g., sending short messages). In the first phase, part of the target sentence appeared on the screen (see Figure 3.3a). Participants were requested to read and enter the sentence part as fast and accurate as possible. Error correction in the first phase was mandatory. Then, participants pressed the "enter" key on the keyboard to trigger the second phase.

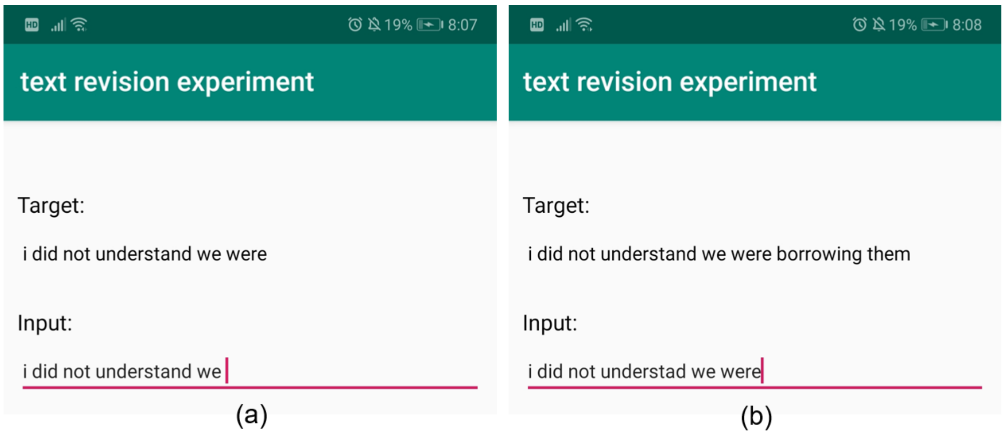


Figure 3.3 The experimental interfaces used for study 1. (a) 1st phase of the trial. (b) 2nd phase of the trial.

In the second phase, the system showed the rest part of the target sentence (see Figure 3b). As instructed in (Arif et al., 2016), we injected an error randomly in the front, middle, or end of the participant's transcription (in real-life scenarios, text revision usually happens unexpectedly during typing). Participants were requested to correct the error and finish the transcription as fast and accurate as possible. After that, participants submitted the input by pressing the enter button (and then started a new trial). The injected error had two main types: character-level and word-level. Each included the following subtypes: insertion, substitution, and omission. The distribution of each type of error is shown in Table 3.1 (distribution of character-level errors followed the statistical results from (Dhakal et al., 2018)).

Table 3.1 Proportions for various types of errors inserted during the experiment

	Character-level	**Word-level**
Insertion	21%	33%
Substitution	53%	33%
Omission	26%	33%

We counterbalanced the order of the techniques across participants using the Latin Square. For investigating the potential learning effect, three blocks of trials were completed by each participant over three consecutive days (with an approximate time gap of more than 24 hours). Every day, participants needed to finish one block of trials for each technique (3 blocks a day). Each block consisted of 24 target sentences randomly chosen from the corpus.

Participants practiced sufficiently with all techniques. During the study, participants were seated in front of the desk in comfortable postures. All participants used two hands to hold the smartphone and typed on the virtual keyboard. Participants got 5-minute rest between blocks. As the result, the total number of trials was:

9 participants × 3 techniques (conventional backspace & caret, magnifier, Dot Swap) × 24 sentences × 1 block/day × 3 days = 1944.

3.6.4 IVs and DVs

The independent variables in this study were *technique* and *error type* (character-level, word-level). We evaluated text revision performance with the following dependent variables: 1) *correction time* (duration between the first action and the final action when revising the target), 2) *number of caret control operations* (for Dot Swap, content navigation was regarded as the caret control operation), 3) *caret control time* (duration of the finger navigating the caret or the content), 4) *number of backspace keystrokes*, and 5) *backspace time* (duration between the previous keystroke and the backspace keystroke). For Dot Swap, the behavior of navigating the content for replacement was also calculated as the number of caret control operations and caret control time.

3.6.5 Results

We did the log-transformation operation and validated the data normality of correction time, caret control time, and backspace time. Further, we performed the repeated measures ANOVA on those DVs ($\alpha = 0.05$, post-hoc tests with Bonferroni correction). For the number of caret control operations and the number of backspace keystrokes, data did not satisfy the normality, thus we performed the Friedman test and Wilcoxon Signed-Rank test.

The average correction time for conventional backspace & caret, magnifier, and Dot Swap (same order hereinafter) were 7158.28ms (*SE* = 323.45), 7068.61ms (*SE* = 313.38), and 6440.65ms (*SE* = 381.81), respectively. An ANOVA showed a significant effect of technique ($F_{2,16} = 8.21$, $p<.001$, $\eta_p^2 = 0.03$), error type ($F_{1,8} = 81.63$, $p<.001$, $\eta_p^2 = 0.21$) and technique × error type ($F_{2,16} = 5.18$, $p<.01$, $\eta_p^2 = 0.02$) on the average correction time. Post-hoc analysis revealed that Dot Swap showed a significant difference ($p<.01$) from the other two techniques.

For the average number of caret control operations, participants navigated the caret most times with the conventional backspace & caret (2.62, *SE* = 0.12), then with the magnifier (2.27, *SE* = 0.08). Dot Swap navigated the caret with the least number (1.29, *SE* = 0.05). The Friedman test showed a larger effect of technique ($\chi^2(2) = 536.05$, $p<.001$) for the number of caret control operations. A post-hoc test using the Wilcoxon Signed-rank tests with Bonferroni correction showed significant differences between the three techniques ($p<.001$).

The average caret control time for three techniques were 1813.29ms (*SE* = 103.72), 1754.12ms (*SE* = 122.8), and 1100.08ms (*SE* = 86.16), respectively. An ANOVA showed a significant effect of technique ($F_{2,16} = 186.72$, $p<.001$, $\eta_p^2 = 0.37$) and error type ($F_{1,8} = 15.88$, $p<.001$, $\eta_p^2 = 0.05$) on the average caret control time. Post-hoc analysis revealed that Dot Swap showed a significant difference ($p<.001$) from the other two techniques. The interaction effect of technique× error type ($F_{2,16} = 2.84$, $p = 0.06$, $\eta_p^2 = 0.01$) was not significant.

For the average number of backspace keystrokes, the conventional backspace & caret used the most 4.19 (*SE* = 0.17), then the magnifier 4.02 (*SE* = 0.17). Dot Swap used the least number of backspace keystrokes 2.44 (*SE* = 0.66) during the revision. The Friedman test showed a larger effect of technique ($\chi^2(2) = 190.14$, $p<.001$) for the number of backspace keystrokes. A post-hoc test using the Wilcoxon Signed-rank tests with Bonferroni correction showed significant differences between the three techniques ($p<.01$).

The average backspace time for three techniques were 542.60ms (*SE* = 38.59), 773.11ms (*SE* = 76.1), and 692.82ms (*SE* = 163.8), respectively. An ANOVA showed a significant effect of technique ($F_{2,16} = 9.67$, $p<.001$, $\eta_p^2 = 0.03$), error type ($F_{1,8} = 84.7$, $p<.001$, $\eta_p^2 = 0.21$) and technique × error type ($F_{2,16} = 51.05$, $p<.001$, $\eta_p^2 = 0.14$) on the average backspace time. Post-hoc analysis revealed that Dot Swap showed a significant difference ($p<.001$) from the other two techniques.

3.6.6 Discussion

Based on the results, Magnifier can help participants to decrease the effort when using backspace and caret. However, it mainly focused on improving the controllability of the caret instead of changing the text revision procedure. When revising character-level errors, all participants commented that instead of navigating the caret, they preferred to enter the whole word.

Though Dot Swap showed an overall reduction in the average correction time, it didn't achieve the improvement we expected. There were three reasons for that. First, participants needed time to memorize and adapt to the rules of using Dot Swap. Therefore, more cognitive effort was taken to decide the correction strategy and following procedures during revisions. Three participants commented that sometimes there were confusions when entering the dot as the dot was often regarded as the ‘period’ symbol indicating the end of a sentence. Second, the text in the text area was small. Thus, due to the finger occlusion, it was difficult to navigate the content for correction towards small targets (1 or 2 characters) with their fingertips. Third, when participants made an unintended correction (to the wrong place) or the intended correction was misspelled, extra time and steps were required to recover from the compounded mistakes. This can lead to cascading errors.

3.7 Improved Swap

Based on results and concerns revealed in study 1, we made the following improvements in the iterative technique (named improved Swap).

First, we used a text buffer (Microsoft, 2018) to remove the ambiguity caused by the identifier in the Dot Swap and make the entered content available for revision. The text buffer is the widget holding the pronunciation spellings in Chinese text entry methods (Microsoft, 2018). The content just entered first goes to the text buffer (see Figure 4). One can either press the “enter” button to make the content appear at the end of the input string or use it for revision. Meanwhile, the text buffer will be emptied for further input. During the implementation, we set the text buffer size as 150 characters to satisfy most of the conditions (on average 70-100 characters for one sentence (Cutts, 2013)).

Second, we designed an expanded layout to make all words and spaces visible and easy to select. In detail, after entering the content for revision, participants can tap the input string to trigger an expanded layout (see Figure 3.4). In the expanded layout, we turned words and spaces near the tap position into buttons. One can press the corresponding button to finish the replacement with the content in the text buffer. After that, the expanded layout disappeared with the caret remained at the end of the input string and the text buffer emptied. If participants triggered the expanded layout in the wrong place, they can press the “CANCEL” button (see Figure 3.4) to revoke the expanded layout. If there was no content in the text buffer when tapping the input string, the expanded layout would not be triggered.

Last, we integrated the symbolized backspace on the virtual keyboard as a button. When participants pressed the symbolized backspace, an emoji (a red cross, see Figure 3.5b) appeared at the end of the text.

In summary, when using improved Swap, participants first entered the content at the end of the text, tapped the target position to trigger the expanded layout, and then pressed the target button in the expanded layout to finish the revision (replacement).

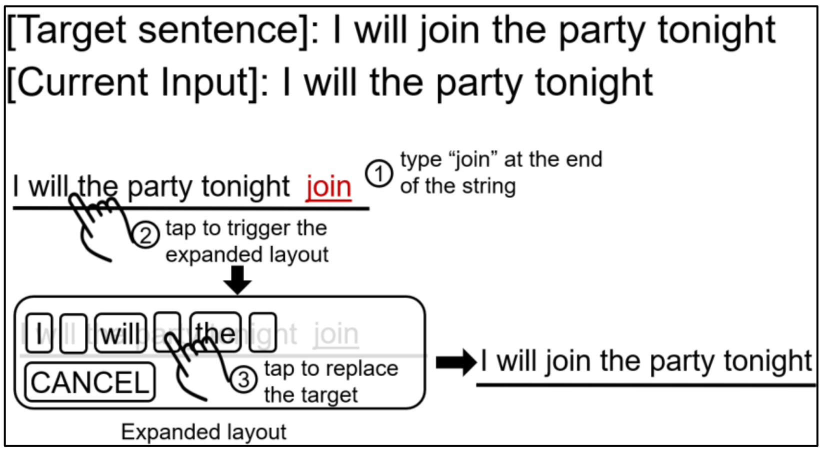

[Target sentence]: I will join the party tonight
[Current Input]: I will the party tonight
I will the party tonight join
① type "join" at the end of the string
② tap to trigger the expanded layout
I will the join
CANCEL
③ tap to replace the target
I will join the party tonight
Expanded layout

Figure 3.4 The schematic of using the improved Swap. The enclosed numbers refer to the operation steps.

3.8 Study 2 - Comparative User Study

We conducted a user study to examine the performance of improved Swap and compared its text revision efficiency with the conventional backspace & caret. In this study, participants were requested to revise paragraphs, which contained multiple words to correct. In addition to the number and time consumption when using the backspace and navigating the caret, we also explored participants' entry speed with different text revision techniques. Furthermore, we collected participants' feedback and preferences regarding those techniques.

3.8.1 Apparatus

We used the same apparatus in Study 1 (with a custom application for the paragraph revision task). Similarly, we also disabled auto-correction and auto-completion.

3.8.2 Participants

Eighteen participants (eight females, age ranged from 21 to 32 years, average 26.33, *SE* = 0.9, one left-handed) participated in the study. Before the experiment, we asked all participants to do simple typing speed tests on their smartphones (https://10fastfingers.com/). The average typing speed was

23.94 WPM (*SE* = 1.79). No participants had physical problems with their hands or eyes. All participants had at least 3 years of experience in using mobile devices. None of the participants was a native English speaker. And, none of them knew the replacement-based text revision techniques before. Each participant received a voucher equivalent to $10 as compensation.

3.8.3 Procedure and Task

To further validate the capacity of the replacement-based text revision process in case of heavy revision conditions, we used a paragraph revision task in this study. For each technique, participants were given six paragraphs (with eight revision targets in each paragraph) to revise. All paragraphs were selected from Wikipedia. The error types and their distribution are shown in Table 3.1.

The study lasted about 70 minutes for each participant. Participants first signed the informed consent form. Then we demonstrated two techniques to participants and asked them to adjust themselves into a comfortable sitting position in front of the desk. Then, participants can get sufficient practice to ensure that they were familiar with the techniques. After that, participants were requested to revise each paragraph with the assigned technique as fast and accurate as possible with two thumbs.

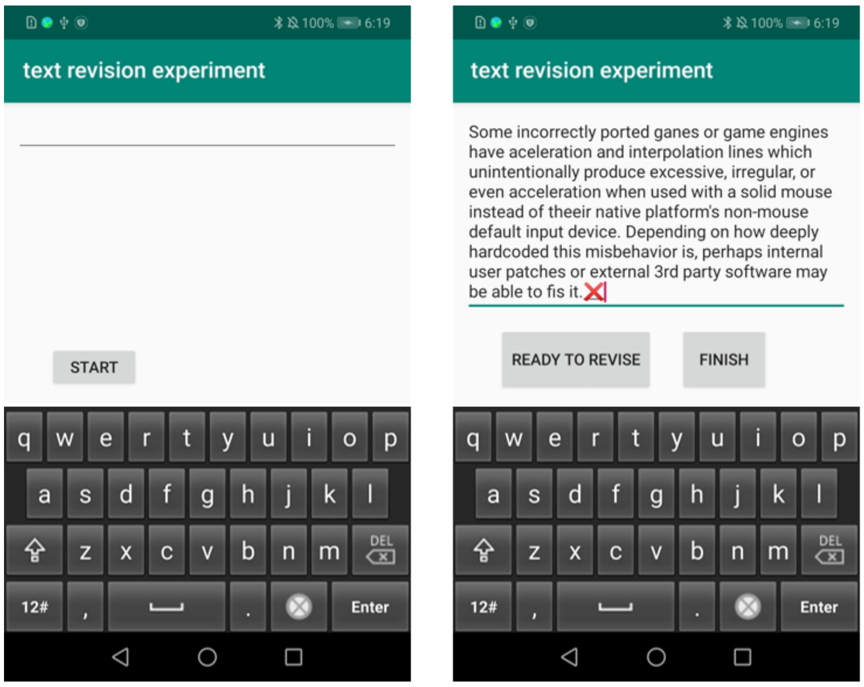
When revising the paragraph, we provided the corresponding printout with all revision targets marked on it to the participant, because 1) it is difficult for them to detect word-level errors (e.g., a word without typos but fails to express the participants' intention) and 2) we mainly focus on investigating the revision efficiency of different techniques, not the participants' text comprehension ability. With marked targets on the printout, participants could reduce the effort of observing targets during text revision.

The experiment began when the participant pressed the "START" button (see Figure 3.5a). Then the paragraph (with revision targets) appeared on the touch screen (see Figure 3.5b). We requested participants to revise all targets. For each target, when the participant observed it and prepared to revise, s/he needed to press the "READY TO REVISE" button and then start the revision. When all targets were corrected, s/he pressed the "FINISH" button to finish the revision of the given paragraph. No help was provided by the experimenter during the experiment.

Participants could get enough rest between paragraphs. After revising all paragraphs, we asked participants to fill in the NASA-TLX (see Appendix 1) and a 7-point Likert Scale questionnaire (1 for the less, 7 for the most, the lower, the better, see Appendix 2) for subjective evaluations on factors including complexity, fatigue, difficulty, and the dislike for the two techniques.

We used the within-subjects design in this study. The order of the paragraphs and the techniques for each participant was totally randomized. The total number of revisions in this study was:

18 participants × 2 techniques (conventional backspace & caret, improved Swap) × 6 paragraphs × 8 revision targets per paragraph = 1728.



Some incorrectly ported ganes or game engines have aceleration and interpolation lines which unintentionally produce excessive, irregular, or even acceleration when used with a solid mouse instead of theeir native platform's non-mouse default input device. Depending on how deeply hardcoded this misbehavior is, perhaps internal user patches or external 3rd party software may be able to fis it.

Figure 3.5 The experiment interfaces for Study 2. (a) is the interface before the experiment. (b) is the interface for the experiment. The symbolized backspace key is on the left side of the "Enter" key. When participants enter the symbolized backspace, a red cross appears on the screen.

3.8.4 IVs and DVs

The independent variables in this study were *technique* (conventional backspace & caret, improved Swap) and *error type*. Error type included character-level ("C") and word-level ("W"). Each had three subtypes: insertion ("i"), substitution ("s"), and omission ("o").

Except for calculating the number of caret control operations, caret control time, number of backspace keystrokes, and backspace time as in Study 1, we also calculated the following metrics:

1) *Keystroke per minute*: the number of keystrokes during the revision process divided by the input time.

2) *Pre-action time*: the time taken to detect the revision target and plan the sequence of actions. It is calculated as the time from the moment that the previous target is revised to the moment the "READY TO REVISE" button is pressed.

3) *Action time*: the time taken to do the actual revision (after pressing the "READY TO REVISE" button). It is the duration between the first and the last action required for the actual revision of the target.

4) *Navigation time*: for improved Swap, it is the total time taken to trigger the expanded layout, press the corresponding button for replacement, and press the "CANCEL" button if the participant mis-triggered the layout; for conventional backspace & caret, it is calculated as the total time taken to control the caret.

5) *Input time*: the time taken to enter characters (including the symbolized backspace).

3.8.5 Results

Sixty-seven (3.88%, sixty-four outliers and three missing data) revision data was removed before the quantitative analysis. We did the log-transformation operation and validated the data normality for DVs in Study 2. Further, we performed the repeated measures ANOVA ($\alpha = 0.05$, post-hoc tests with Bonferroni correction). For the number of caret control operations and the number of backspace keystrokes, data did not satisfy the normality, thus we performed the Wilcoxon Signed-Rank test (also for the subjective evaluation) and Friedman test.

Keystroke per Minute. For each correction, the average keystroke per minute for conventional backspace & caret and improved Swap were 87.43 (*SE* = 1.39) and 124.55 (*SE* = 2.91), respectively. An ANOVA identified a significant effect of technique ($F_{1,17} = 116.07$, $p<.001$, $\eta_p^2 = 0.55$), error type ($F_{5,85} = 6.79$, $p<.001$, $\eta_p^2 = 0.07$), and their interaction ($F_{5,85} = 76.15$, $p<.001$, $\eta_p^2 = 0.44$) on average keystroke per minute.

Time Consumption During Correction. For each correction, the average pre-action time for conventional backspace & caret and improved Swap were 3616.31 (*SE* = 125.77) and 3041.51 (*SE*

= 154.66) ms, respectively. An ANOVA identified a significant effect of technique ($F_{1,17}$ = 18.16, p<.001, η_p^2 = 0.16) on the average pre-action time. The effect of error type ($F_{5,85}$ = 2.08, p = 0.07, η_p^2 = 0.02) and technique × error type ($F_{5,85}$ = 1.17, p = 0.32, η_p^2 = 0.01) was not significant. Figure 3.6 illustrates the details.

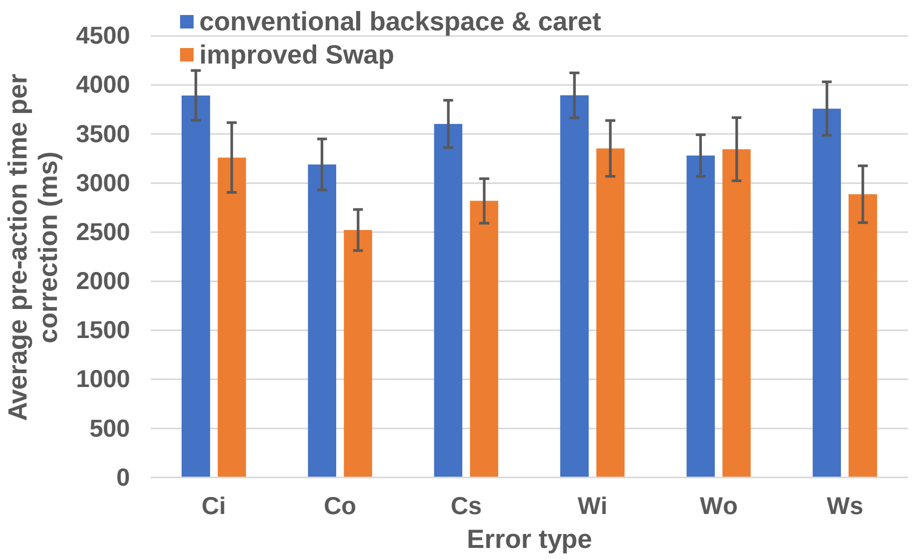


Figure 3.6 Average pre-action time for different error types.

For each correction, the average action time for conventional backspace & caret and improved Swap were 7419.79 (*SE* = 143.4) and 6342.55 (*SE* = 145.99) ms, respectively. An ANOVA identified a significant effect of technique ($F_{1,17}$ = 27.53, p<.001, η_p^2 = 0.22), error type ($F_{5,85}$ = 24.44, p<.001, η_p^2 = 0.2) and their interaction ($F_{5,85}$ = 21.47, p<.001, η_p^2 = 0.18) on the average action time. Pairwise comparison revealed that all three word-level errors were different from each other (all p<.001). There was no significant difference among three character-level errors. Ws also showed significant differences from all other error types (all p<.001). See Figure 3.7 for details.

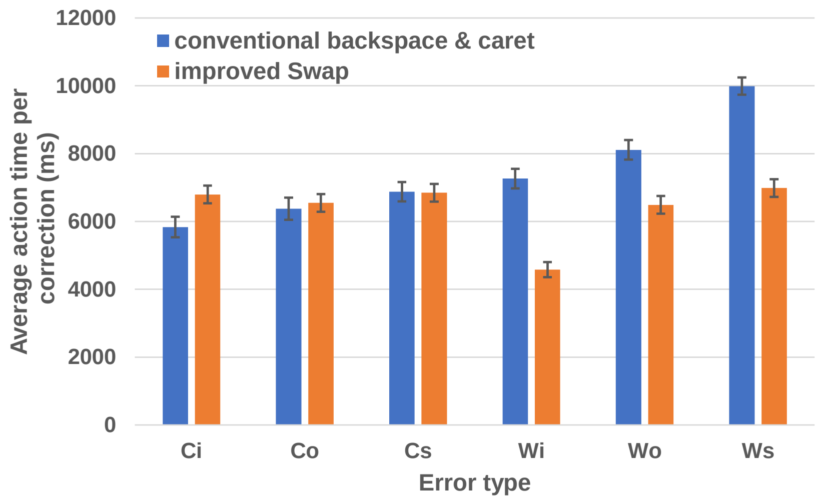


Figure 3.7 Average action time for different error types.

For each correction, the average navigation time and input time for conventional backspace & caret and improved Swap are shown in Figure 3.8. No significant effect of technique was found on average navigation time ($F_{1,17} = 2.21$, $p = 0.14$, $\eta_p^2 = 0.02$). An ANOVA found a significant effect for error type on average navigation time ($F_{5,85} = 8.13$, $p<.001$, $\eta_p^2 = 0.08$). For average input time, an ANOVA identified a significant effect of technique ($F_{1,17} = 125.63$, $p<.001$, $\eta_p^2 = 0.57$), error type ($F_{5,85} = 43.4$, $p<.001$, $\eta_p^2 = 0.31$) and their interaction ($F_{5,85} = 37.64$, $p<.001$, $\eta_p^2 = 0.28$) on the average input time. Correcting the Ws targets with conventional backspace & caret took the longest average input time (*M* = 6677.75ms, *SE* = 196.5), and while correcting the Wi targets, the improved Swap took the shortest average input time (*M* = 1289.4ms, *SE* = 83.27).

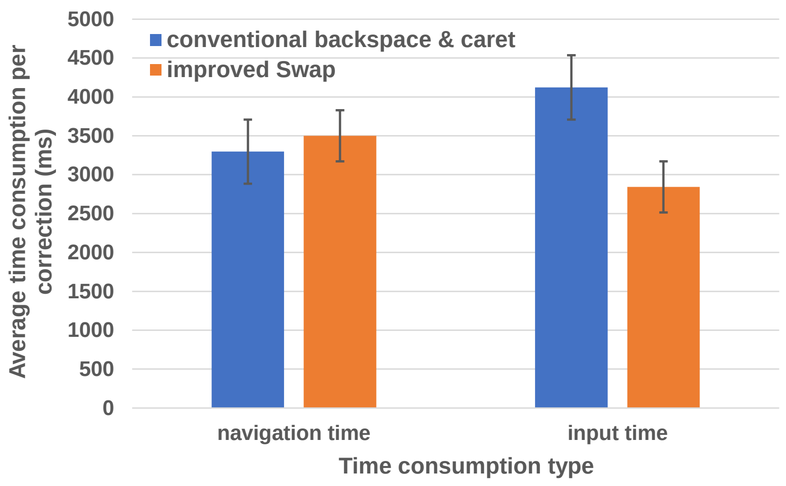


Figure 3.8 Average navigation time and input time for two text revision techniques.

Caret Control. For each correction, the average number of caret control operations for conventional backspace & caret and improved Swap were 2.61 (*SE* = 0.1) and 1.04 (*SE* = 0.01). Figure 3.9 shows the average number of caret control operations per correction for each error type. The Wilcoxon Signed-rank test showed a significant effect among techniques (*W* = 16, *Z* = -18.65, *p*<.001, *r* = 0.46). The Friedman test showed a significant effect among error types ($\chi^2(5) = 34.01$, *p*<.001) for the number of caret control operations. A post-hoc test using the Wilcoxon Signed-rank tests with Bonferroni correction showed significant differences between Ci and Cs, Co and Cs, and Cs and Ws (*p*<.001).

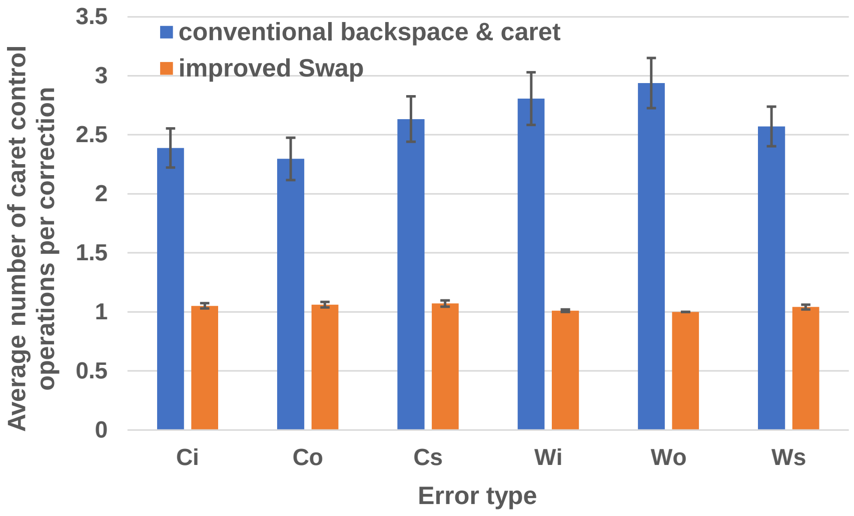


Figure 3.9 Average number of caret control operations for different error types.

For each correction, the average caret control time for conventional backspace & caret and improved Swap were 3297 (*SE* = 102.09) and 2345.62 (*SE* = 75.66) ms. Figure 3.10 shows average caret control time when correcting different types of errors. An ANOVA identified a significant effect of technique ($F_{1,17} = 56.49$, *p*<.001, $\eta_p^2 = 0.37$) and error type ($F_{5,85} = 6.21$, *p*<.001, $\eta_p^2 = 0.06$) on the average caret control time. However, the interaction effect was not significant ($F_{5,85} = 1.65$, *p* = 0.15, $\eta_p^2 = 0.02$). Pairwise comparisons identified the significant difference between Co and Ci (*p*<.05), Wi (*p*<.05), Wo (*p*<.001), and Ws (*p*<.01).

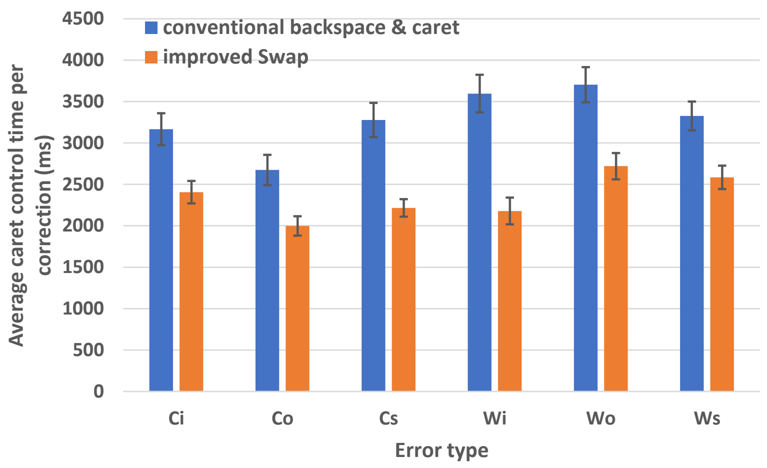


Figure 3.10 Average caret control time for different error types.

The Use of Backspace. For each correction, the average number of backspace keystrokes for conventional backspace & caret and improved Swap were 3.24 (*SE* = 0.08) and 0.08 (*SE* = 0.02) respectively. The Wilcoxon Signed-rank test showed a significant effect among techniques (*W* = 17, *Z* = -21.75, *p*<.001, *r* = 0.54). The Friedman test showed a significant effect among error types ($\chi^2(5)$ = 445.46, *p*<.001) for the number of backspace keystrokes. A post-hoc test using the Wilcoxon Signed-rank tests with Bonferroni correction showed significant differences between all error type pairs (*p*<.01).

The average backspace time for each correction with conventional backspace & caret and improved Swap were 1993.17 (*SE* = 41.73) and 76.83 (*SE* = 16.75) ms respectively. It indicated that participants used fewer backspace keystrokes when entering the content at the end of the paragraph. Pairwise comparison identified a significant difference among word-level errors (all *p*<.001).

Subjective Evaluations. For the weighted NASA-TLX rating scores (the lower, the better), improved Swap (*M* = 45.3, *SE* = 2.7) scored lower than the conventional backspace & caret (*M* = 53.37, *SE* = 3.84). However, a Wilcoxon Signed-Rank test did not find the significant difference between two techniques (*W* = 6, *Z* = 6, *p* = .05, *r* = 0.33). For all factors measured in the Likert Scale (see Figure 3.11), improved Swap got lower scores than conventional backspace & caret. A Wilcoxon Signed-Rank test revealed that, for improved Swap, the score for fatigue (*W* = 2, *Z* = -2.99, *p* <.01, *r* = 0.5), difficulty (*W* = 1, *Z* = -3.22, *p*<.01, *r* = 0.54), and dislike (*W* = 1, *Z* = -3.03, *p*<.01, *r* = 0.51) was significantly lower than conventional backspace & caret.

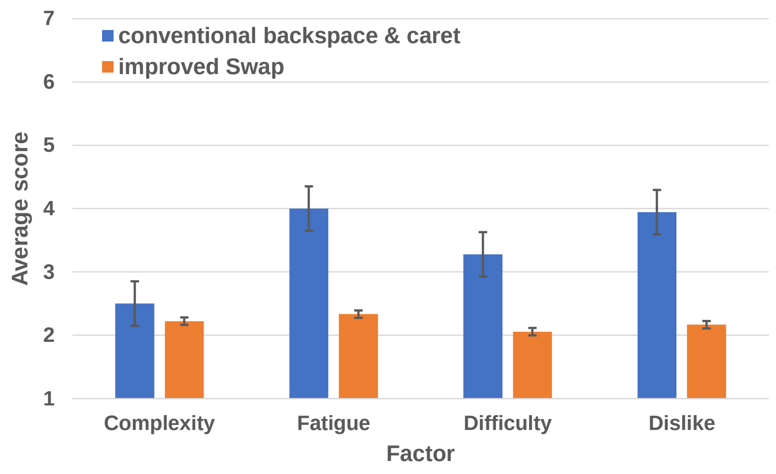


Figure 3.11 Average Likert Scale scores for each factor.

Overall, most participants gave positive responses to improved Swap (P3: "It is easy to understand the logic of the new technique and to get used to it quickly.", P5: "I prefer to use the new technique because it frees me from using the backspace multiple times."). Eight participants described the experience of using improved Swap with the word "intuitive" (P11: "The new technique liberates me from controlling the small caret. I can quickly type the content, locate, and finish the revision."). Five participants pointed out that improved Swap showed advantages, especially when revising word-level targets in long paragraphs. Six participants commented that, in a short period of time, it was hard to change to the new technique due to the years of experience with backspace and caret. Participants regarded that a long-term use of improved Swap would change their habits of using backspace and caret. All participants agreed that improved Swap could decrease the effort required by navigating the caret and pressing the backspace repetitively. Six participants expect to integrate improved Swap into their smartphones.

3.8.6 Discussion

Based on results, improved Swap took less action time when correcting word-level errors, whereas it didn't show advantages when correcting character-level errors. The reason is that improved Swap did the replacement only in word-level. Thus, participants needed to type the whole word and then replace the target. Participants expressed their attitudes towards the character-level revision performance with improved Swap: "*For character-level errors, if the word length is within 5 characters, it would be fine to type the whole word and do the replacement. If the length exceeds that, it might cost extra time for revision. Despite that, I still tend to type because it is easier than navigating the caret.*"

In Study 2, we found that participants adapted two strategies to navigate the caret. One strategy was to persevere in locating the caret with repetitive caret control attempts until it arrived at the intended position. The other strategy was that when the caret was located before the intended position, the participant would re-navigate the caret; if the caret was located 2-3 characters after the intended position, the participant would use the backspace multiple times to delete to the intended place and then perform the revision. One participant gave a reason for adapting the second strategy: "*I usually avoid navigating the caret on the smartphone as the caret is too difficult to locate. Therefore, I would rather spend time deleting and retyping the content than control the caret multiple times.*" The expanded layout made it easier for participants to replace the word as 1) it enlarged the target size for easier selection and 2) it made the replacement intuitive and easy to understand.

Figure 3.8 showed that improved Swap took longer navigation time than the conventional backspace & caret. There are two reasons for that: 1) after typing the content for replacement, it still needs some time for participants to locate the target; 2) it is challenging for participants to fully accept a new technique in a short period of time. Though participants reported that they did sufficient practice, they still need time and effort to get used to the replacement operation.

With improved Swap, participants can type faster than the conventional backspace & caret during the text revision. The average keystroke per minute for improved Swap was similar to the average typing speed for all participants measured before the study. This indicated that improved Swap could leverage participants' regular typing speed to finish the text entry quickly during the revision.

We calculated the total time (sum of pre-action time and action time) to evaluate the time consumption when revising a target. Results proved that, improved Swap took less total time (M = 9195.47ms, SE = 133.71) than the conventional backspace & caret (M = 10654.01ms, SE = 114.38). For word-level errors, it took on average 9248ms (SE = 189.85) to finish the revision, while on average 11688.42ms (SE = 152.05) for the conventional backspace & caret. The reason for the difference was that imporved Swap minimized the use of backspace and caret, and thus decreased the time consumption.

Figure 3.9 revealed that participants were more likely to navigate the caret multiple times when using conventional backspace & caret for revision compared with improved Swap. Through the data, we found 499 revisions navigated the caret more than once with the conventional backspace & caret. For improved Swap, only 35 revisions were found to trigger the expanded layout more than once. It indicated that selecting a target in the expanded layout was easier than navigating the small caret.

3.9 Conclusion

In this chapter, we have presented Swap, a replacement-based technique to facilitate text revision on mobile devices. To simplify the text revision interaction, Swap designed the symbolized backspace and the replacement-based revision process. Swap allowed users to input the content first and then use it for revision. We implemented two techniques (Dot Swap and improved Swap) and compared their text revision efficiency with the conventional text revision techniques through user studies. Results revealed that Swap improves revision efficiency and fluency by significantly reducing the use of backspace and caret during the text revision. Most participants showed their preference for Swap as it is easy to learn, and it is more efficient. Some participants expect to integrate Swap with their own mobile devices for long-term use.

CHAPTER 4

DESIGN SPACE EXPLORATION ON TEXT REVISION IN VR APPLICATIONS

Current VR systems provide various text input methods that enable users to enter text efficiently with virtual keyboards. However, little attention has been paid to facilitate text revision during the VR text input process. We first summarized existing text revision solutions in current VR text input research and found that backspace is the only tool available for text revision with virtual keyboards with few mentioning designs for caret control. To systematically explore VR text revision designs, we presented a design space for VR text revision based on backspace and caret. With the proposed design space, we further analyzed the feasibility of the combined usage of backspace and caret by proposing and evaluating four VR text revision techniques. Outcomes of this research can provide a fundamental understanding of VR text revision solutions (with backspace and caret) and a comparable basis for evaluating future VR text revision techniques.

4.1 Introduction

The consumer-affordable and increasingly popular VR devices not only offer users the immersive and engaging experience but also provide a solid hardware basis for the development of productive working environments such as virtual office work (Grubert, Ofek, et al., 2018; Guo et al., 2019) and remote classrooms (C. Ma et al., 2009). In such scenarios, users often need to deal with text content (e.g., sending emails, composing reports, taking class notes). Thus, effective text input methods play a vital role to ensure both quality and satisfaction when users interact in such immersive virtual environments. These text input methods, besides providing rapid typing speed, should also be capable of handling various conditions such as typo (and grammar) correction and content rephrasing.

Compared to physical keyboards, virtual keyboards are more suitable (Speicher et al., 2018) and portable (Dube & Arif, 2019) to deploy in VR applications with less environmental requirements (e.g., stable and flat surface to place the physical keyboard). Numerous studies have endeavored to enhance typing speed and typo correction with virtual keyboards by proposing novel typing methods and auto-correction algorithms. However, although text revision is a vital subtask to ensure the accuracy of the input content (Li et al., 2020), it has not yet received sufficient attention in the current VR text input research. As an example, we can imagine a scenario of composing an email in a VR system: after quickly entering (typo-free) text, the user still needs to revise the content (e.g., deleting, adding, or substituting words) to make sure that it can express the intention accurately and meaningfully. Currently, backspace on the virtual keyboard is the dominant tool for revision. However, we found that most backspace functions mentioned in current VR text input research can only be activated at the end of the input stream. In practical typing scenarios, revision targets can appear anywhere in the input content, which means that current VR text revision solutions cannot handle comprehensive revision requirements effectively.

To further understand the status quo of the VR text revision, we investigated existing text revision tools applied in current VR typing techniques and found two issues: 1) most researchers intend to facilitate typing speed with higher accuracy (typo-free) but with little consideration on how to improve text revision efficiency, and 2) caret control is not included in current VR virtual keyboard-based text input designs; backspace is the only tool available to satisfy the basic need of text revision without considering efficiency. To better leverage backspace and caret in VR text revision, we first provide a design space to explore the possibilities of applying various backspace-caret designs. Based on the proposed design space, we then implemented four VR text revision techniques with virtual keyboards and handheld controllers. We evaluated their text revision performance through a comparative user study. Results show that combining word-level deletion and continuous caret control achieves better text revision performance in VR. Finally, we further discuss findings and summarize them into general guidelines for the future design of VR text revision tools and techniques.

The contributions of this work are as follows. First, we reveal the gap in the VR text revision area with an extensive overview of the current VR text input designs. Second, we provide a design space for better exploration of VR text revision designs based on the backspace and caret. Third, we propose and evaluate four VR text revision techniques with a comparative study. Fourth, we provide design guidelines based on the experiment results and findings in order to enable researchers to explore novel tools (and techniques) beyond backspace and caret or to introduce solutions applied in other platforms (e.g., smartphones) into VR text revision designs.

4.2 Related Work

Text revision is a ubiquitous and vital subtask during text input activities (Li et al., 2020). To accomplish text revision, users need to use various tools such as caret and backspace to insert, delete, or substitute words to revise part(s) of the content. In this section, we reviewed VR text input techniques and summarized their attempts to facilitate text revision (see Table 4.1 and Table 4.2 as the overview).

4.2.1 Error correction and text revision

Numerous researchers have endeavored to eliminate typos and grammar issues with en/decoders (Levenshtein, 1966; C. Yu et al., 2017), auto-prediction, and auto-completion algorithms. Those techniques help users to ensure the typing quality at the level of spelling and grammar. Their work provides us with a solid foundation to discuss more general text input scenarios in the wild. Compared with transcription in lab studies, typing in real-life conditions not only needs to consider the spelling and grammar accuracy but also needs to make sure that the text users type expresses their intention accurately and clearly (Li et al., 2020). Beyond the "mothering" of (semi-) automatic error correction techniques, there still exist requirements to revise text. Generally, when composing emails (whether formal or informal), typo-free is just the basic criterion. More importantly, users need to make sure all items written in the email accurately express nuanced information with an appropriate expression for stakeholders. In some cases (e.g., business email), it requires minor modifications through rounds of proofreading and adjustment. However, for such conditions, there lack efficient tools for text revision beyond error (typo) correction.

4.2.2 Text revision with physical devices in VR

Physical keyboards (with QWERTY layout) are powerful devices used for VR text input. Due to the occlusion of Head-Mounted Displays (HMDs), typing with physical keyboards in VR cannot achieve equivalent performance as that in the real world. Thus, a series of research was conducted to investigate factors such as hand representation (Grubert, Witzani, et al., 2018a; Knierim et al., 2018), keyboard representation (Mcgill et al., 2015), and blending of reality in VR (Lin et al., 2017) and their influence on users' typing performance. As the default components of physical keyboards, backspace and arrow keys are available to accomplish text revision (including error correction). Although arrow keys were referred to in previous designs (Bovet et al., 2018; Grubert, Witzani, et al., 2018b; Hoppe et al., 2018; Knierim et al., 2020; Menzner et al., 2019; Otte, Menzner, et al., 2019; Otte, Schneider, et al., 2019; Pham & Stuerzlinger, 2019), no mention is made regarding their use during the typing task or how the use of arrow keys may potentially influence users' VR typing (and revision) performance. Walker et al. (Walker et al., 2017) disabled the backspace key and provided an auto-correction algorithm as an alternative in order to avoid the influence of user differences on data quality.

Other devices such as data gloves and custom hardware are also used for VR text entry. Pinch Keyboard (D. A Bowman et al., 2001; Doug A. Bowman et al., 2002; González et al., 2009) allows users to enter text via hand rotation and pinches while wearing data gloves. Pinch Keyboard allows users to perform a backspace with a pinch gesture using two ring fingers. However, there is no caret control design in their technique. KITTY (Kuester et al., 2006; Mehring et al., 2004) maps keys to finger joints. Users enter text by pinching different parts of the finger with the thumb. KITTY did not include caret control in its design neither. While they mentioned the existence of the delete key, no clear statement was found regarding how to correct typos. Wu et al. (Wu et al., 2017) used data gloves to provide haptic feedback when pressing the VR keyboard. Their design allowed users to use backspace and arrow keys to revise text. K3 (Brun et al., 2019) is a cubic object that allows designers to integrate buttons on it for various interactions (including text input). As it is mainly for developing a new interaction platform, K3 did not mention the details of its VR text input design.

4.2.3 Text revision with virtual keyboards in VR

Compared with physical keyboards, virtual keyboards in VR merely exploit backspace as the dominant tool for revising text content. Surprisingly, to the best of our knowledge (according to the summary of Table 2), no current virtual keyboards include caret control solutions in their designs, and no research proactively discusses caret control with virtual keyboards in VR. Basically, backspace in the majority of virtual keyboards (Boletsis & Kongsvik, 2019; Boustila et al., 2019; Chen et al., 2019; Choi et al., 2019; Dash, 2017; Fashimpaur et al., 2020; Gugenheimer et al., 2016; Ishii et al., 2017; Jiang & Weng, 2020; Jimenez & Schulze, 2018; Kim & Kim, 2017; Lu et al., 2019; Min, 2011; Ogitani et al., 2018; Prätorius et al., 2014, 2015; Rajanna & Hansen, 2018; Son et al., 2019; Speicher et al., 2018; Wilson & Agrawala, 2006; Yanagihara & Shizuki, 2018; D. Yu et al., 2018) follows the function inherited from physical keyboards: erase one character before the caret and move the caret backward (Wikipedia, 2019). Backspace can be effective for quick corrections

(Komninos et al., 2018) (e.g., delete/change the typo character near the caret (Li et al., 2020)). However, for targets far from the caret, it is challenging to finish the revision in VR due to the limitation of character-level backspace and the lack of caret control.

To finish the revision, users need to trigger the backspace multiple times. During the deletion, part of the correct text will also be deleted, which requires more time and effort to recover. Arif et al. (Arif et al., 2016) addressed this issue in the context of error correction on smartphones by leveraging the Levenshtein distance (Levenshtein, 1966) (to locate the error position) and gestures (swiping left or right on the backspace key to cut off or recover the content after the error position). However, for typo-free content, Levenshtein distance does not function well, making it difficult to find the error position.

Researchers also updated the backspace function to achieve word-level deletion. RotoSwype (Gupta et al., 2019) and GestureType (C. Yu et al., 2017) allow users to delete the word just entered. RingText (Xu et al., 2019), DwellType (C. Yu et al., 2017), and TapType (C. Yu et al., 2017) enable flexible deletion (at the character-level when entering and at the word-level when editing) according to the content just entered. The effort mentioned above improves the deletion efficiency of backspace. Without caret control solutions, however, flexible backspace may only show satisfying revision performance at the end of the input stream.

Table 4.1 Physical VR text input techniques and their design details

Keyboard and Layout	Brief Description	Backspace	Caret Control	Text Revision
Physical_Qwertz (Grubert, Witzani, et al., 2018b)	Type on a physical keyboard/touch screen with/without a positioned view	Yes	Arrow keys	CBs[b]
Physical_Qwerty (Pham & Stuerzlinger, 2019)	Type on a physical keyboard placed on a hawker tray	Yes	Disabled	CBs[b]
Physical_Qwerty (Otte, Menzner, et al., 2019)	Type on a touch-sensitive physical keyboard	Yes	Arrow keys	CBs[b]
Physical_Qwerty (Knierim et al., 2020)	Type on a wireless keyboard with a portable VR HMD on the users' head	Yes	Arrow keys	NM
Physical_Qwerty (Hoppe et al., 2018)	Type on a physical keyboard with a real-world representation in the HMD	Yes	Arrow keys	NM
Physical_Qwerty (Bovet et al., 2018)	Type on a physical keyboard with a 3D keyboard model shown in HMD	Yes	Arrow keys	NM
Physical_Qwerty (Walker et al., 2017)	Type on a physical keyboard with key pressing visual feedback in HMD	Disabled	NM	NM
DataGlove_Qwerty (D. A Bowman et al., 2001; Doug A. Bowman et al., 2002; González et al., 2009)	Users select characters with hand rotation and confirm with a pinch	Yes	No	NM
DataGlove_Qwerty (Wu et al., 2017)	Users press virtual keys by bending the finger over the angle threshold	Yes	Arrow keys	NM
DataGlove_Qwerty (Kuester et al., 2006; Mehring et al., 2004)	Users select characters by pinching thumb with joints on other fingers	Yes	No	NM
Cubic_NM[a] (Brun et al., 2019)	Users hold the cube and press buttons on it to enter characters	NM	NM	NM

[a] NM represents "not mentioned". [b] CBs represents "character-level backspace".

4.2.4 Summary

As one of the practical activities in everyday life, text input is not a simple task to press multiple buttons on keyboards but usually to record ideas, reach agreements, or express emotions. Typing speed and accuracy are indeed key factors to contribute to better typing behavior, but they are not decisive factors (Dube & Arif, 2019). Currently, text input techniques can help users to enter text quickly into computers with few typos and grammar issues. It should be noted that text input has two parts: entering text into the computer system and taking care of the text already in the system. However, for current VR text input techniques with virtual keyboards, backspace is the only tool to feed the basic need for text revision (without discussing efficiency and satisfaction). Backspace and arrow keys can be easily implemented in VR scenarios, and users can transfer their experience using physical backspace and arrow keys to VR scenarios (Hoppe et al., 2018). However, there is a lack of essential consideration regarding text revision and adjusted solutions in current VR text input designs with virtual keyboards.

4.3 Design Space

To fill the gap of text revision in current VR text input designs, we propose a design space for a structural understanding and exploration of VR text revision solutions. Such a design space can assist practitioners and researchers in proposing text input techniques that aim more at practical use in VR scenarios. Two parameters that constitute the design space are backspace granularity and caret control continuity (as shown in Figure 4.1). Each parameter has two values: backspace granularity (*character-level* or *word-level*) and caret control continuity (*discrete* or *continuous*). In this 2 x 2 matrix, each combination represents a design solution using various types of backspace and caret control for VR text revision.

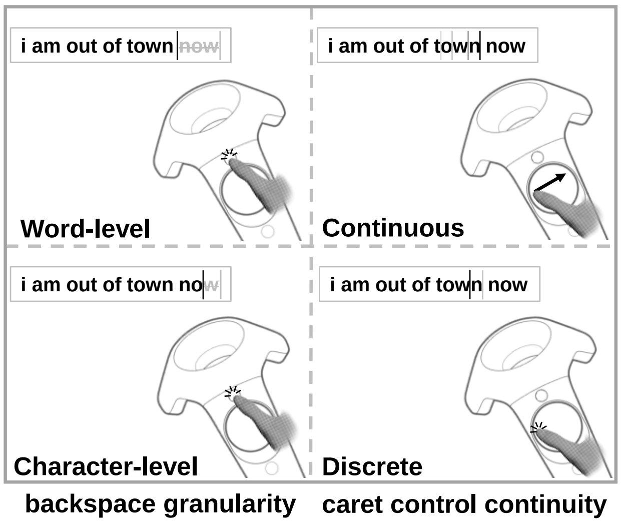


Figure 4.1 Different options for backspace granularity and caret control continuity.

4.3.1 Backspace granularity

Backspace granularity is divided into *character-level* and *word-level*. We choose these two values based on the summary of current VR text input techniques (see Table 4.1 and Table 4.2 for details). Although few references clearly stated the use of backspace for text revision (most mentioned it for typo corrections), it is reasonable to infer that backspace is capable of handling both typo correction and text revision. Character-level is the default granularity for backspace which means that only one character will be deleted when users press/select the backspace key once. Word-level implies that one press of the backspace key can erase multiple characters.

4.3.2 Caret control continuity

Caret control continuity is divided into *discrete* and *continuous*. This division is inspired by the descriptions of various approaches to navigate the cursor to select keys in selection-based VR text input techniques (Speicher et al., 2018) and users' movements when performing actions (Janzen et al., 2014). Discrete caret control means that when users control the caret, one single operation (e.g., button pressing) can move the caret once. If users need to move the caret away (e.g., by a distance of five characters), they have to repeat the operation multiple times. For continuous caret control, users can navigate the caret with a single smooth movement rather than by repetitive keystrokes or gestures.

Table 4.2 Virtual VR text input techniques and their design details

Keyboard and Layout	Brief Description	Backspace	Caret Control	Text Revision
Virtual_Qwerty (Rajanna & Hansen, 2018)	Users gaze at the character and confirm it with dwell or button click	Yes	No	NM[a]
Virtual_ Qwerty (C. Yu et al., 2017)	(TapType and DwellType) Users rotate head to point at a letter and dwell/press button to confirm (GestureType) Users use eye-movement to perform a word-level gesture on the keyboard	Yes	No	CBs[b], WBs[c]
Virtual_ Qwerty (Speicher et al., 2018)	(Head Pointing) Users point at the letter with head and confirm with a button click (Controller Pointing) Users point at the letter with handheld controllers and confirm with a button click (Controller Tapping) Users use the handheld controller to poke letters (Freehand) Users use a finger to poke letters (Discrete Cursor) Users press the touchpad to move the cursor and select letters via trigger (Continous Cursor) Users swipe on the touchpad to move the cursor and select letters via trigger	Yes	No	NM
Virtual_ Qwerty (Ishii et al., 2017)	Users move a thumb-up hand to point at a letter and select it with fist posture	Yes	No	NM
Virtual_3x3 layout (Min, 2011)	Users press the key with the intended character multiple times to finish the input	Yes	No	NM
Virtual_ Qwerty (Gugenheimer et al., 2016)	Users press the letter on a touchscreen on HMD and confirm it with finger liftoff	Yes	No	NM
Virtual_ Qwerty (Kim & Kim, 2017)	Users hover the finger on the letter and select it by pressing down	Yes	No	NM
Virtual_A~Z layout (X. Ma et al., 2018)	Users select letter through the analysis of gaze data and EEG signals from the brain	No	No	NM
Virtual_ Qwerty (Jimenez & Schulze, 2018)	Users move the cursor via head/pointer finger movement and interact with keys via hand gestures	Yes	No	NM
Virtual_Sector (D. Yu et al., 2018)	Users use the left joystick to choose the intended slice and the right joystick to choose the letter	Yes	No	NM
Virtual_ Qwerty (Wilson & Agrawala, 2006)	Users select the letter with the left/right thumbstick and confirm it with trigger buttons	Yes	No	NM
Virtual_3x3x3 spatial layout (Yanagihara & Shizuki, 2018)	Users select characters by drawing a stroke among spatial buttons with trigger pressing on a controller and confirm them with a trigger release	Yes	No	NM
Virtual_A~Z layout (Dash, 2017)	Users select the character by pinching between the thumb and parts of other fingers	Yes	No	NM
Virtual_12-Key (Prätorius et al., 2014, 2015)	Users select a character by pinching between the thumb and the part of other fingers multiple times	Yes	No	NM
Virtual_12-Key (Ogitani et al., 2018)	Users select a character by first pressing the button and then swipe the direction of the intended character	Yes	No	NM
Virtual_ Qwerty (Boustila et al., 2019)	Users control the cursor to the letter with the finger on the touchscreen and release the finger to confirm	Yes	No	NM
Virtual_ Qwerty (J. Dudley et al., 2019)	Users type on a virtual keyboard in the mid-air with bare hands with tracking markers	No	No	NM
Virtual_ Qwerty (Chen et al., 2019)	Users select characters by performing word-level gestures with controllers or on touchscreens	Yes	No	NM
Virtual_ Qwerty (Choi et al., 2019)	Users type on a virtual keyboard on a surface with bare hands with tracking markers	Yes	No	NM
Virtual_ Qwerty (Gupta et al., 2019)	Users select characters by performing word-level gestures with a ring worn on the index finger	Yes	No	WBs
Virtual_ Qwerty (Boletsis & Kongsvik, 2019)	(Raycasting) Users use rays emitted from controllers to point at the letter and confirm it with button pressing (Drum-like keyboard) Users use virtual mallets to strike the virtual keys to input characters (Head-directed input) Users rotate head to point at the letter and confirm it with button pressing (Split keyboard) Users move fingers on controllers' touchpad to move cursors and press the touchpad for selection	Yes	No	NM
Virtual_Circular layout (Xu et al., 2019)	Users select characters with head rotations among characters and auto-prediction word candidates	Yes	No	CBs, WBs
Virtual_Circular layout (Jiang & Weng, 2020)	Users press keys on the outer part of the controller touchpad. Then press the center touchpad to select the intended word	Yes	No	CBs
Virtual_ Qwerty (Lu et al., 2019)	Users rotate the head to point at the letter and confirm it by moving the head forward and backward	Yes	No	NM
Virtual_ Qwerty (Son et al., 2019)	Users do slide-and-click or fly-and-touch on two touchpads to enter the character	Yes	No	NM
Virtual_ Qwerty (Fashimpaur et al., 2020)	Users enter characters by the thumb to fingertip pinches	Yes	No	NM

[a] NM represents "not mentioned". [b] CBs represents "character-level backspace". [c] WBs represents "word-level backspace".

4.3.3 *Functionality of the design space*

The proposed design space provides combinations that integrate values selected from both backspace granularity and caret control continuity. Inspired by Hirzle et al. (Hirzle et al., 2019), we discuss the effects and applicability of the proposed design space for VR text revision solutions.

Currently, most physical VR text input techniques follow the conventional character-level backspace and caret control solution that is already applied on physical keyboards. However, current techniques with virtual keyboards cannot be included in this design space as they do not incorporate caret control into their designs. Without caret control, revision can only happen at the end of the input stream with 1) repetitive backspace selections and 2) extra deletion and cascade recovery input (for revisions far from the caret). For text revision with virtual keyboards, the proposed design space brings caret control back into design considerations. This also allows us to combine backspace and caret control features and properties to achieve better VR text revision performance.

4.4 Text Revision Techniques

To further investigate how users manipulate the backspace and caret to finish text revision, this section illustrates four VR text revision design combinations and the techniques that exploit each of them. Among various input devices (e.g., data gloves, eye trackers, stylus, or bare hands) used in VR systems, we choose the handheld controller to implement VR text revision techniques for the following reasons. First, handheld controllers provide a highly portable platform with rich interaction diversities. Second, using handheld controllers (gesture-based interaction excluded) can imply less cognitive and motor load to users compared with bare-hand gestures and head-based movement. Last but not least, it is a cost-friendly option for broad deployment. For concise descriptions, we mention the combinations of various types of backspace and caret (see Figure 4.1) with the following abbreviations: **CBs-DCc** for character-level backspace & discrete caret control, **CBs-CCc** for character-level backspace & continuous caret control, **WBs-DCc** for word-level backspace & discrete caret control, and **WBs-CCc** for word-level backspace & continuous caret control.

4.4.1 *CBs-DCc*

CBs-DCc is the most fundamental combination inherited from the backspace key and arrow keys on physical keyboards. In previous designs, backspace can be selected by either poking (Speicher et al., 2018), striking (Boletsis & Kongsvik, 2019), or pressing buttons on handheld controllers (Jiang &

Weng, 2020). Although there is no reference mentioning the caret control, we got inspiration from (Speicher et al., 2018), which mentioned that directional pads or thumbsticks could be used to move the cursor discretely. In the technique of *CBs-DCc*, users can navigate the caret by pressing four directions of the touchpad. To delete text, users can perform the character-level deletion by pressing the button on handheld controllers.

4.4.2 CBs-CCc

CBs-CCc shares the same idea with the combined-use of the physical keyboard and touchpad on a laptop. With the finger sliding on the touchpad, users can move the cursor in a smooth and continuous way. Compared with discrete caret control, the continuous operation allows users to navigate the caret with one single movement instead of repetitive operations. In the technique of *CBs-CCc*, we use the same character-level deletion as in *CBs-DCc*. For caret control, we leverage touchpads on HTC Vive (HTC Corporation, 2015) handheld controllers to achieve continuous caret control. Users can perform the sliding movement on the touchpad to navigate the caret. Real-time visual feedback was provided to track the caret position while sliding on the touchpad. When the finger is lifted to disengage from the touchpad, the caret remains at the last finger contact position.

4.4.3 WBs-DCc

WBs-DCc extends the function of conventional backspace to multi-character deletion. For revision scenarios where there is a need to delete word(s) with improper meaning (Li et al., 2020), word-level backspace could, to some extent, show more efficiency than character-level backspace. In the technique of *WBs-DCc*, users can press the button on the handheld controller to delete multiple characters according to the caret position in the text. *WBs-DCc* follows the design of *CBs-DCc* for caret control by pressing directional parts of the touchpad.

4.4.4 WBs-CCc

WBs-CCc combines multi-character deletion and continuous caret control for text revision. When using the *WBs-CCc* technique to revise the text, users can first navigate the caret to the position of the revision target by sliding the finger on the touchpad. Users can then execute the revision with multi-character deletion by pressing the button on the handheld controllers.

4.4.5 Features of proposed techniques

Here we discuss characteristics of those text revision techniques before conducting the comparative study for further evaluations. First, all proposed text revision techniques depend on a virtual keyboard with the Qwerty layout as the text revision also includes the text entry process (e.g., substituting or inserting words). Second, compared with conventional VR text input techniques, four proposed VR text revision techniques enable the controllability of the textbox widget for caret manipulation. Third, all techniques use handheld controllers (mainly with the touchpad and buttons) to realize the deletion and caret control. In fact, the orientation of the controller itself can also be leveraged for caret control. For instance, using the virtual ray emitted from the controller head to navigate the caret among characters. However, due to users' poor depth perception towards targets in VR (Chan et al., 2010), it is challenging for users to perform ray-casting target selection and control, especially for targets with limited size (Rizzo, 2019; Susu et al., 2019), let alone the caret with the width of a few pixels. To implement ray-casting interaction in practical VR text revision designs, various factors (e.g., target size and distance) should be further confirmed and investigated via a series of systematic empirical studies, which are beyond the scope of this study.

4.5 Comparative Evaluation

To further evaluate VR text revision performance with the four text revision techniques based on virtual keyboards and handheld controllers, we conducted a within-subjects experiment with 16 participants to compare the proposed VR text revision techniques: *CBs-DCc*, *CBs-CCc*, *WBs-DCc*, and *WBs-CCc*.

4.5.1 Participants

16 participants (9 female, 7 male) aged between 22 and 28 years (*M* = 24.81, *SD* = 2.4) volunteered for this study. All participants are physically healthy with either normal vision or corrected to normal vision with glasses. Only one participant reported the dominant hand as the left hand. None of the participants had prior VR text input experience. As reported by participants, all of them are familiar with the Qwerty keyboard and can read and comprehend English sentences.

4.5.2 Apparatus

We built an experimental system using the HTC Vive (including the HMD, two optical trackers, and two default handheld controllers) and a desktop computer with an intel i7-3770 CPU, 16GB RAM, and NVIDIA Quadro K4000 graphic card. Optical trackers are set 2 meters above ground to detect the motion that happened in an area of 2.5m × 2.5m. We chose to use Drum-like keyboard (GoogleDaydream, 2016) as the text input technique in the experiment because it showed reasonable typing speed with less error, less frustration, and high user preferences, as reported in the literature (Boletsis & Kongsvik, 2019). Drum-like keyboard allows users to enter text by striking keys on the virtual keyboard with virtual mallets stretching from the head of handheld controllers. Drum-like keyboard and handheld controllers also helped us to focus on the text revision analysis with less adverse influence from other input methods (e.g., mid-air virtual keyboard with bare hands). The experiment system was implemented based on Cutie keys (Cutiekeys, 2017) and ran on the Windows 10 operating system with Unity 5.6. We learned from the word deletion solution (Raymond, 2007) applied in current computer operating systems to implement the word-level backspace.

4.5.3 Corpus and revision targets

We chose 120 sentences (with an average length of 6.56 words (31.1 characters) per sentence) from the Enron Mobile Email Dataset (Vertanen & Kristensson, 2011) to build the experimental corpus because, 1) all sentences are easy to remember and easy to comprehend as they are all selected from daily life use, 2) its validity has been examined in prior VR text input studies (J. Dudley et al., 2019; Speicher et al., 2018), and 3) compared with the phrase set from MacKenzie and Soukoreff (MacKenzie & Soukoreff, 2003), sentences in the Enron Dataset are longer, which is easy for us to arrange revision targets. All chosen sentences do not contain any uppercase letters, numbers, or punctuations.

We included only one revision target for each sentence. We learned from the categorization of character-level error types (Gentner et al., 1983) and extended it to word-level text revision: omission (a word is missing), insertion (there is an extra word), and substitution (a wrong word appears in place of the intended one), as in (Li et al., 2020). For evaluating the text revision capability of those techniques towards targets in different positions, we also defined half of the targets for each type that are within the last three words away from the end of the sentence, while the other half of targets far from the end of sentences, as in (Zhang et al., 2019). On average, each target contains 5.08 characters (*SD* = 1.79).

4.5.4 Design and procedure

The experiment used a within-subjects design to evaluate participants' VR text revision performance using the four proposed techniques. The independent variable was *text revision technique* with four values (*CBs-DCc*, *CBs-CCc*, *WBs-DCc,* and *WBs-CCc*). Every participant needed to use four text revision techniques to finish the sentence revision task (30 revisions for each technique). We counterbalanced the order of text revision techniques among participants with a Latin Square. As a result, the total number of revision targets was: 16 participants × 4 text revision techniques × 30 sentences per technique = 1920. We randomly chose every 30 sentences from the corpus and ensured that those sentences did not appear in other sentences. For every 30 sentences, the appearance order of all types of targets was randomized (see Table 4.3 for the detailed distribution).

Table 4.3 Number of different types of revision targets for every 30 sentences

Target type	Omission	Insertion	Substitution	Total
Far	5	5	5	15
Near	5	5	5	15
Total	10	10	10	30

We first introduced the purpose of the study and demonstrated the four text revision techniques and various types of revision targets to all participants. Then, we guided participants to sit on a chair in the middle of the tracking area and helped them to put on the HMD and hold the two controllers. After adjusting the height and orientation of the virtual keyboard, all participants had a 15-minute practice to get used to the Drum-like keyboard and various text revision techniques with demo sentences before the formal experiment. Participants were recommended to use the touchpad and buttons on the controller in their dominant hand for revision. Revision trials were designed as follows. Two sentences (one is the standard sentence, the other is the sentence with a revision target) first appeared in the experimental system. Participants then used the assigned technique to revise the target and pressed the grip button on the tail part of the controller to submit the revision and to start the next trial. One trial can be submitted successfully only if two sentences were matched after the revision.

The formal experiment lasted around 60 minutes. Participants were requested to finish the text revision task with the assigned technique as fast and accurate as possible. After revising 30 sentences, participants could have a 5-min break before the next 30 sentences with another text revision technique. After all revisions, we asked participants to fill the NASA-TLX (Hart & Staveland, 1988) and System Usability Scale (SUS, see Appendix 3) (Brooke, 2013) to evaluate workload and subjective preferences towards all four techniques.

4.5.5 Results

To simulate the real-life text revision scenario, we mixed various types of revision targets and showed them to participants in random order, which is different from lab studies that artificially provide targets with the same type multiple times. Therefore, we mainly investigated and reported the influence of technique on participants' VR text revision performance. We removed 92 items of data (4.79%, including outliers and trials interrupted by the occasional trigger of the VR system menu) from the dataset. As the dataset did not pass the normality check, we used the Friedman test and the post-hoc using Wilcoxon signed-rank test with Bonferroni correction. For subjective evaluation, we reported weighted TLX scores (Hart, 2006) and analyzed SUS scores using the Friedman test. Error bars shown in the following figures represent the standard error, and data labels represent the mean values.

Operation per character stands for the average number of operations (including character, backspace and caret control) required to revise one character during the revision. The operations per character of four proposed techniques (in ascending order) were *WBs-CCc* (*M* = 1.52, *SE* = 0.33), *CBs-CCc* (*M* = 2.06, *SE* = 0.32), *WBs-DCc* (*M* = 4.26, *SE* = 0.82) and *CBs-DCc* (*M* = 4.81, *SE* = 0.89). A Friedman test revealed a significant effect of technique on the operation per character ($\chi^2(3) = 682.68$, $p < .001$). The post-hoc showed the significant effect of technique between all technique pairs (all $p < .001$). Figure 4.2 shows the operation per character when participants revised targets with different distances.

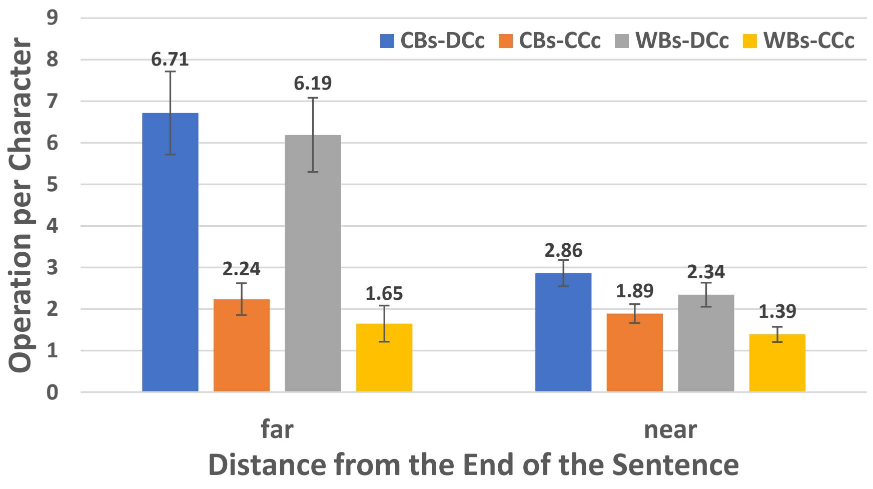


Figure 4.2 Operation per character for targets far from or near the end of the sentence.

Correction time denotes the average duration required to revise the given targets. It is calculated as the interval between the start of a trial and the moment when participants submit the revision. The correction time of the four proposed techniques (in ascending order) were *WBs-CCc* (*M* = 7978.19 ms, *SE* = 726.22), *CBs-CCc* (*M* = 8771.53 ms, *SE* = 730.62), *WBs-DCc* (*M* = 9062.8 ms, *SE* = 719.85) and *CBs-DCc* (*M* = 9269.94 ms, *SE* = 702.99). A Friedman test revealed a significant effect of technique on the correction time ($\chi^2(3) = 335.9$, $p < .001$). The post-hoc showed the significant effect of technique between all technique pairs (all $p < .001$). Figure 4.3 shows the correction time when participants revised targets with different distances.

Backspace frequency and Caret frequency count the average number of performing the backspace and caret navigation operation for each revision. *WBs-CCc* (*M* = 1.04, *SE* = 0.21) used the least number of backspace while *CBs-CCc* (*M* = 3.59, *SE* = 0.7) used the most. A Friedman test revealed a significant effect of technique on the backspace frequency ($\chi^2(3) = 252.08$, $p < .001$). The post-hoc did not find the significant effect of technique between two CBs-based techniques ($p = 0.61$, ns) or between two WBs-based techniques ($p = 0.53$, ns). Figure 4.4 shows the backspace frequency when participants revised targets with different distances.

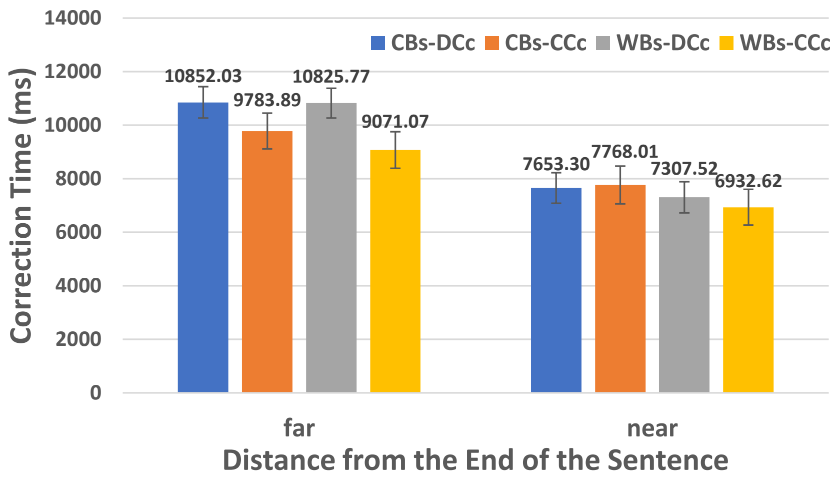


Figure 4.3 Correction time for targets far from or near the end of the sentence.

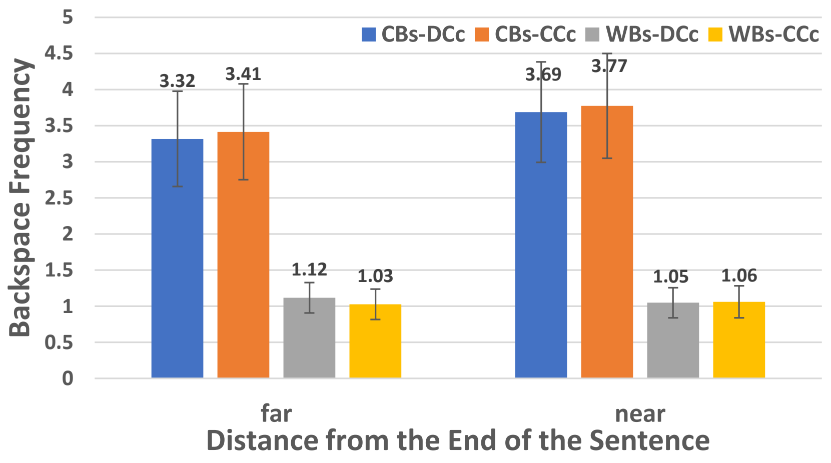


Figure 4.4 Backspace frequency for targets far from or near the end of the sentence.

For caret control, *CBs-CCc* (*M* = 2.32, *SE* = 0.54) used the least number of caret control while *CBs-DCc* (*M* = 13.59, *SE* = 1.86) used the most. A Friedman test revealed a significant effect of technique on the caret frequency ($\chi^2(3) = 1013.75$, $p < .001$). The post-hoc did not find the significant effect of technique between two DCc-based techniques ($p = 0.2$, ns) or between two CCc-based techniques ($p = 0.62$, ns). Figure 4.5 shows the caret frequency when participants revised targets with different distances.

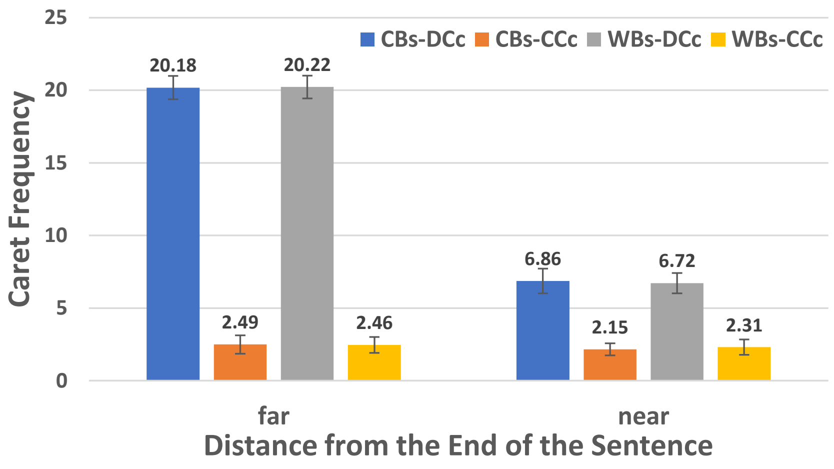


Figure 4.5 Caret frequency for targets far from or near the end of the sentence.

Caret control time stands for the total time spent navigating the caret in each revision. *WBs-CCc* (*M* = 2120.62 ms, *SE* = 315.85) showed the least caret control time followed with *CBs-CCc* (*M* = 2194.47 ms, *SE* = 325.67), *CBs-DCc* (*M* = 5118.88 ms, *SE* = 540.56), and *WBs-DCc* (*M* = 5361.48, *SE* = 539.48). A Friedman test revealed a significant effect of technique on the caret control time ($\chi^2(3) = 755.29$, $p < .001$). The post-hoc showed a significant effect (all $p < .05$) of technique between all technique pairs except between *CBs-CCc* and *WBs-CCc* (p = 0.68, ns). Figure 4.6 shows the caret control time when participants revised targets with different distances.

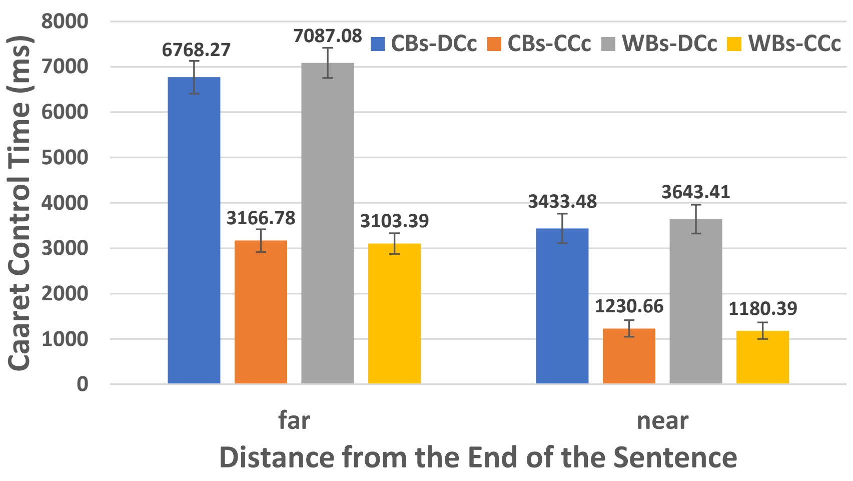


Figure 4.6 Caret control time for targets far from or near the end of the sentence.

Backspace time denotes the average total time spent pressing the backspace when revising each target. It is calculated as the total duration between the previous keystroke and the backspace. *WBs-CCc* (M = 775.11 ms, SE = 166.11) showed the least backspace time followed with *WBs-DCc* (M = 940.61 ms, SE = 192.89), *CBs-CCc* (M = 1622.54 ms, SE = 327.78) and *CBs-DCc* (M = 1630.81 ms, SE = 320.54). A Friedman test revealed a significant effect of technique on the backspace time ($\chi^2(3)$ = 196.14, $p < .001$). The post-hoc showed a significant effect (all $p < .001$) of technique between all technique pairs except between *CBs-DCc* and *CBs-CCc* (p = 0.74, ns). Figure 4.7 shows the backspace time when participants revised targets with different distances.

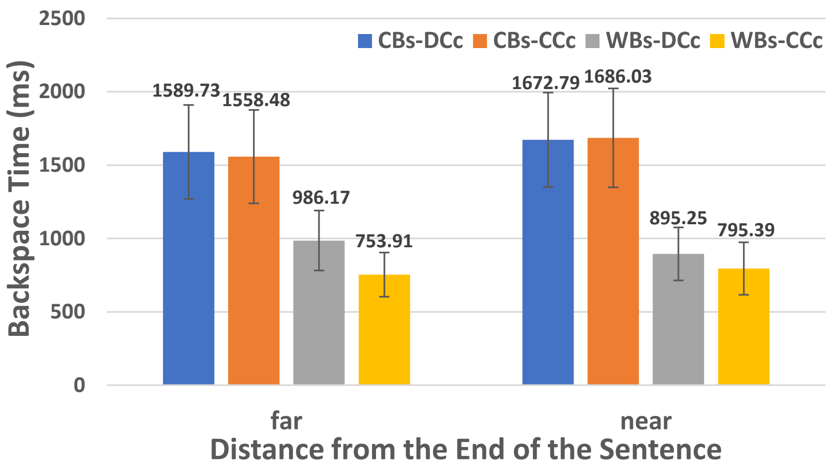


Figure 4.7 Backspace time for targets far from or near the end of the sentence.

In the paired-scale weighting process of NASA-TLX, the top three scales participants scored were Effort (63), Performance (57) and Physical demand (46). *WBs-CCc* got the lowest (M = 41.89) weighted scores (lower score means less load to finish the task) followed with *WBs-DCc* (M = 52.06), *CBs-CCc* (M = 56.23) and *CBs-DCc* (M = 70.89). For SUS, *WBs-CCc* got the highest score (M = 71.41, SE = 1.84) while *CBs-DCc* got the lowest (M = 64.69, SE = 1.84). A Friedman test revealed a significant effect of technique on the SUS score ($\chi^2(3) = 15.53$, $p < .01$). The post-hoc showed the significant effect of technique between *CBs-DCc* and *WBs-DCc* (Z = -2.93, p <.01, r = 0.74), *CBs-DCc* and *WBs-CCc* (Z = -2.9, p <.01, r = 0.73), and between *CBs-CCc* and *WBs-CCc* (Z = -2.39, p <.05, r = 0.6).

4.6 Discussion

In this section, we first discuss findings according to the quantitative results and participants' feedback. Then, we summarize and propose a series of design guidelines for the future development and implementation of VR text revision tools and techniques. Finally, we list a series of future work for in-depth research on text revision in VR environments.

We re-introduced caret control to VR text revision in this study and evaluated participants' text revision performance with four backspace-caret techniques. Overall, results showed that WBs-CCc outperformed other proposed techniques with the least operation number per character during revision, the shortest correction time, including the lowest number of operations for caret control and backspace. Participants also regarded that using WBs-CCc can finish the revision with less workload and high usability.

Results from the backspace frequency and backspace time analysis indicated that word-level backspace frees participants from repetitive backspace pressing operations during revision. We did not include the recovery function in the word-level backspace design. It should be noted that if participants unintentionally delete multiple words or mis-delete other words, extra time and effort would be required to recover from the mis-operation. Participants also commented that they needed extra effort to re-evaluate the sentence after larger changes were made (e.g., deleting a whole word). Moreover, it is inevitable to enter typo characters during revision. In that case, there are two solutions. One is to use the word-level deletion to delete multiple characters and re-enter them. The other is to use the character-level backspace on the virtual keyboard (we did not disable the backspace function on the virtual keyboard). It was interesting to observe from the log files that if the typo was observed immediately (usually 1-2 characters away from the caret), participants attempted to use the backspace on the virtual keyboard for quick character-level deletions, or they chose to use the word-level backspace. From the considerations above, we inferred that 1) assistive mechanisms should be included to deal with exceptions using word-level backspace, and 2) character-level backspace is still needed for quick deletion. After the formal experiment, we asked five participants to continue the revision task with another 30 sentences using only virtual keyboards (without caret control). After the revision, we asked them about their preferences between the backspace on the controller and the backspace on the virtual keyboard. They showed their preference towards backspace on the controller. This is because pressing the physical button only uses muscles around the thumb while striking the backspace on the virtual keyboard uses the fore-arm with a higher physical workload.

In contrast with conventional non-caret designs, caret control made it flexible to navigate the caret to the revision target and thus saved time and avoided extra effort for massive deletions and re-entry (due to the fixed caret position at the end of the input string). Results (Figure 4.2 and Figure 4.5) showed that using continuous caret control can decrease the number of operations during revisions, especially for targets far from the end of the sentence. However, it is a double-edged sword. Ideally, participants can perform the continuous caret control once, and then the caret would be located as intended. However, Figure 5 did not support such an idealized condition. The reason is that when sliding on the touchpad, the caret moves quickly towards the revision target and participants need to track the position and lift their thumbs to finish the caret control. Sometimes the caret would stop before or after the intended position (inconsistent with expectations) after a long-time caret movement. Moreover, there exist possibilities of the subtle finger sliding offset when lifting the thumb from the touchpad.

NASA-TLX results show Effort, Performance, and Physical demand as the top three scales. Such three scales provide us with the design criterion from users' perspectives. In detail, to finish VR text revision tasks with satisfying performance, the technique should have the capacity to handle various exceptions and make sure the revision can be completed successfully with the less physical and mental workload.

4.7 Conclusion

Efficient VR text input techniques provide the potential to move real-life typing activities into the virtual world. However, there would still be far from practical use if text input techniques are designed merely around typing speed and accuracy. After entering text into the systems with few typos, users also need to revise them for expression and description accuracy. In this section, we first conducted a thorough literature review and summary to reveal the status quo of current VR text input designs regarding text revision. The summary revealed a lack of essential considerations on enhancing users' text revision performance in current VR text input designs. Especially for text input with virtual keyboards, only the backspace is available for text revision without mentioning the efficiency. To fill this gap, we provided a design space in VR text input for further exploration of the combined use of backspace and caret control in VR text revision. Based on the design space, we implemented four text revision techniques with virtual keyboards and handheld controllers and evaluated their performance with a comparative study. Results of the study could serve as the groundwork and reference for the future design of VR text revision techniques. We make the initial step to fill the gap of text revision facilitation in VR text input. We believe this work will also raise awareness and interest towards text revision improvements when designing VR text input techniques for practical use.

CHAPTER 5

FACILITATING VR TEXT ENTRY WITH DAILY-LIFE METAPHOR

This chapter introduces the design and validation of the novel VR text entry technique. In detail, we present SewTyping, a novel technique to facilitate controller-based text entry performance in virtual reality (VR) applications. We learned from real-life sewing behavior and achieved successive character selection during the text entry process. We make all buttons on the virtual keyboard interactive on both the top and bottom sides. When entering text with the handheld controller, characters can be selected alternately by penetrating down from one key and up to another. SewTyping enhances the fluidity of text entry with successive sewing-like movements to reduce pauses among character button selections. We ran a comparative user study to evaluate SewTyping with Controller Pointing and Drum-like Keyboard. Results revealed that: 1) SewTyping achieved 25.94 words per minute (WPM) with a total error rate of 4.19% and 2) users adapted to the sewing-like movement easily and achieved steady performance with essential practice.

5.1 Introduction

Recent market trends show that VR techniques have become popular and affordable among consumers (360marketupdates, 2019). VR also shows great capabilities and potential in applications such as education (C. Ma et al., 2009), remote collaboration (Nguyen et al., 2017), and office work (Grubert, Ofek, et al., 2018; Guo et al., 2019). Efficient and effective text entry methods play a vital role in those applications (Dube & Arif, 2019). Although virtual keyboards and related techniques can satisfy the essential need of text entry in VR, these tend to be slow and error-prone (Dube & Arif, 2019) due to 1) the penetrable feature (Chan et al., 2010) of intangible displays (e.g., handheld controllers penetrate the virtual surface inevitably when hitting the virtual target or walking through a wall unexpectedly in VR games (Ogawa et al., 2020)) and 2) the user's insufficient capacity to determine the z-coordinate of a target in VR (Chan et al., 2010).

When entering text in VR with virtual keyboards, most VR text entry methods require two steps to enter a character. Users first locate the intended key and then finish the selection with a confirmation mechanism such as dwell (C. Yu et al., 2017), button pressing on controllers (Speicher et al., 2018), or direct touch (J. J. Dudley et al., 2018). To enter multiple characters, users need to finish the aforementioned process repetitively, which leads to pauses between characters and influences the fluidity of text entry.

We propose SewTyping, a technique to enhance text entry performance with virtual keyboards and handheld controllers in VR applications. SewTyping leverages the penetrable feature in VR to improve the fluidity of text entry. This feature makes all keys (on the virtual keyboard) selectable from both the top and bottom sides. With this change, when users select one key, the extra penetrated movement can be leveraged directly for selecting the following letter. As shown in Figure 5.1, with the handheld controller (and a virtual mallet extended from the head of the controller), users can enter multiple characters with a single and fluid movement on the virtual keyboard, just like sewing fabric with a needle in daily life.

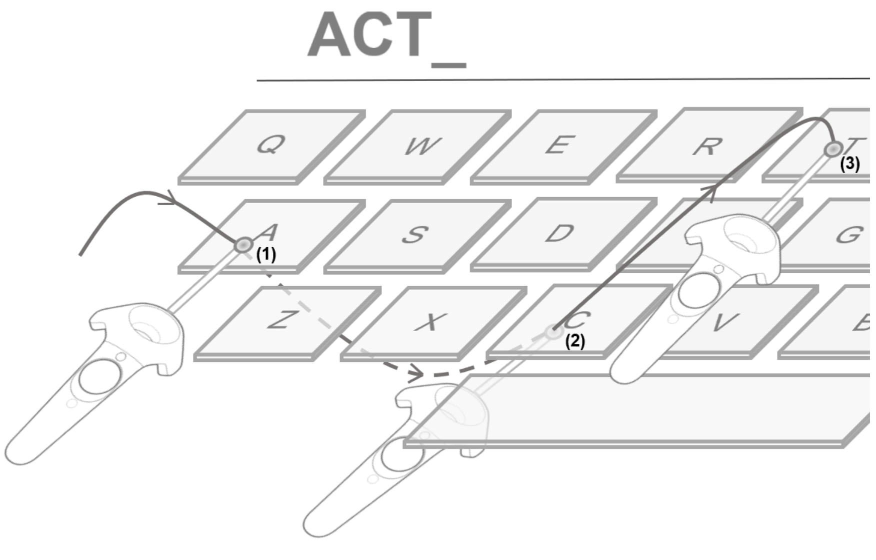


Figure 5.1 Text entry with SewTyping. Characters can be selected with the handheld controller (with a virtual mallet on the top) successively by penetrating alternately on either the top or bottom side of virtual keys. E.g., to enter "ACT", "A" and "T" are selected from the top side, while "C" is selected from the bottom side.

In the following parts of this chapter, we first summarize current VR text entry methods with virtual keyboards. Then we illustrate the design of SewTyping followed by describing a conducted text entry task in VR to explore and validate the feasibility of SewTyping. Results revealed that, with SewTyping, users can achieve 25.94 WPM with an error rate of 4.19%. Users adapted to the sewing-like movement easily by associating it with the sewing metaphor. They also agreed that SewTyping makes VR text entry interesting and engaging.

5.2 Related Work

Physical and virtual keyboards are widely used for text entry in VR. Compared with physical keyboards, virtual keyboards can be deployed flexibly with fewer spatial limitations. Generally, virtual keyboards could not achieve satisfying typing speed as with physical keyboards (Dube & Arif, 2019). Thus, in this section, we mainly review existing VR text entry techniques based on virtual keyboards and identify their design divergences and similarities.

5.2.1 Hand, Head, and Eye-based Text Entry Techniques

A considerable number of studies focused on proposing VR text entry techniques with users' hands. FistPointer (Ishii et al., 2017) allows users to perform a thumb-up gesture to move the cursor and a fist gesture to select a particular character. Speicher et al (Speicher et al., 2018) proposed a technique that allows users to poke the virtual keyboard directly with their fingers (detected by Leap Motion). Ogitani et al. (Ogitani et al., 2018) made a 12-key keyboard based on the Japanese smartphone keyboard and projected it on either the palm or in mid-air. Users point the button and swipe towards the intended character. DigiTap (Prätorius et al., 2014) provides a 12-key keyboard and maps keys on knuckles and finger tips. Similarly, BlueTap (GooglePatent, 2017) maps all letters on knuckles and fingertips singly in alphabetical order. Users tap the button once (GooglePatent, 2017; Prätorius et al., 2014) or multiple times (Prätorius et al., 2014) with the thumb to select the intended letter. Time and effort are required to learn and master these techniques (including new layouts). Evaluations reveal that they cannot provide promising typing speed (no faster than 10 WPM (Dube & Arif, 2019)).

Yu et al. (C. Yu et al., 2017) used head movement to navigate the cursor and select the character by either dwelling on it or pressing a physical button (similar to (Speicher et al., 2018)). Yu et al. (C. Yu et al., 2017) also designed a word-level technique (named GestureType). Users perform a gesture with their head movement (while pressing and holding the button) to select characters and finish the selection by releasing the button. Although GestureType achieved 24.7 WPM, its frequent head movement also brings high physical load to users and may cause discomfort (e.g., dizziness) and therefore limit the potential for its wide and long-term use. Rajanna and Hansen (Rajanna & Hansen, 2018) used eye-tracking data to point at the character (on the virtual keyboard) which is then selected by either a button click or dwelling. Ma et al. (X. Ma et al., 2018) navigated the cursor and selected characters by analyzing real-time eye-tracking data and brain signals collected from the user. However, these gaze-based techniques cannot achieve promising typing speed.

5.2.2 Controller-based Text Entry Techniques

Current VR products (e.g., HTC Vive and Oculus) are equipped with handheld controllers as the default input devices. Cubic keyboard (Yanagihara & Shizuki, 2018) arranges all the letters into a 3 × 3 × 3 spatial layout. Users first press and hold a button on the controller to trigger the keyboard; they then select characters by navigating the controller to reach each of the characters; and finally, they finish the selection by releasing the button. Speicher et al. (Speicher et al., 2018) proposed four techniques with the handheld controller: the first one uses the controller tail to poke the key directly; the second one moves the controller (with the ray that is emitted from it) to point at the key and confirms the selection by pressing the trigger button; the third and the fourth navigate the cursor by either pressing or sliding on the controller trackpad and then confirming the selection by pressing the trigger button.

Some techniques used handheld controllers based on real-life metaphors. Drum keys (GoogleDaydream, 2016) regard the keyboard as a drum and use controllers as virtual mallets to select characters by striking the keys. Boletsis and Kongsvik (Boletsis & Kongsvik, 2019) conducted a comparative study among four VR text entry techniques (Raycasting, Drum-like Keyboard, Head-directed input, and Split keyboard). Results showed that the Drum-like Keyboard outperformed other techniques with 21.01 WPM on average.

Min et al. (Min, 2011) integrated all letters and other characters into the 3 × 3 layout. Users point at the key with a pointing device and then select the character by pressing the button once or multiple times. PizzaText (D. Yu et al., 2018) arranges all characters into a 7-slice circle layout. One uses the left thumbstick of a game controller to select the slice and the right thumbstick to select the intended character. Wilson and Agrawala (Wilson & Agrawala, 2006) moved the joystick on the game controller to highlight the character and finished the selection by pressing the trigger button on the controller. Chen et al. (Chen et al., 2019) implemented a smartphone keyboard in the VR environment. Similar to (C. Yu et al., 2017), users can perform a word-level gesture with either the beam emitted from the handheld controller or the finger on the smartphone screen.

5.2.3 Summary

Overall, current VR text entry techniques with virtual keyboards (and handheld controllers) cannot match the typing speed of physical keyboards (as reported in (Dube & Arif, 2019), mainly between 24.3 and 68.5 WPM). There are three reasons for that: 1) due to the inadequate ability to determine the z-coordinate of the target (Chan et al., 2010), users may inevitably penetrate the keyboard and consume extra time and effort recovering after selecting each character; 2) many text entry techniques (Ishii et al., 2017; Kim & Kim, 2017; Min, 2011; Rajanna & Hansen, 2018; Speicher et al., 2018; Wilson & Agrawala, 2006; C. Yu et al., 2017; D. Yu et al., 2018) require extra steps to confirm the selection after navigating the cursor to the intended character, and this tends to consume excessive time and movement; 3) most techniques regard characters as the basic input unit. Users need to input characters individually and repetitively according to their character selection method which may cause pauses among characters and thus reduce the fluidity of text entry. Although some gesture-based (word-level) techniques could promote typing speed, they either add extra physical burden (Dube & Arif, 2019) or influence the flexibility of handling independent characters (e.g., error correction) while performing gestures. In fact, speech recognition can achieve much faster typing performance than keyboard-based typing when sending short messages (Ruan et al., 2018) and this can be easily implemented in VR scenarios. Whereas, compared with keyboard-based typing, voice input techniques encounter the contradiction that it requires a quieter environment and may bring disturbance to others when using it in a quiet environment. Although voice input techniques also use virtual keyboards to enter text (e.g., revising text), most of the text entry work is finished by speech recognition algorithms rather than the keyboard. Therefore, we didn’t include voice input in our considerations.

5.3 Sewing-like Interaction

Inspired by the real-life sewing metaphor, we designed the sewing-like interaction, which more efficiently achieves successive target selection in VR environments. Based on the sewing-like interaction, we design SewTyping to facilitate VR text entry performance using virtual keyboards and handheld controllers.

5.3.1 Interaction Pattern

To simulate the real-life sewing movement in VR, we first make all virtual targets interactive from both the top and the bottom sides of virtual targets. Users navigate the handheld controller towards the virtual target and finish the selection by hitting either the top or the bottom side of them. Selections will be confirmed as the penetration occurs.

Then, learning from the Drum-like Keyboard (Boletsis & Kongsvik, 2019), we extend a virtual mallet (with a small sphere on the tip) from the top of the controller because it is difficult to select the target (especially in the limited size) using the whole body of the handheld controller. Users move the handheld controller and use the virtual mallet to perform target penetration.

Double-side interactive targets and the handheld controller with a virtual mallet make it possible to realize the sewing-like interaction. After hitting one target, there is no need for a backward movement (see the dashed part in Figure 5.2a). Instead, as shown in Figure 5.2b, users can move the controller directly to the following target and finish the selection. In this case, all targets will be "sewed" together with a single and fluid movement rather than repetitive up-and-down movements.

5.3.2 Features

Generally, existing target selection methods in VR are variants of those methods applied in the physical world. For instance, hitting virtual targets for selection simulates the pressing process on physical buttons; using the ray (e.g., emitted from the handheld controller) to navigate the cursor and confirming the selection by button pressing simulate the use of a mouse in graphical user interfaces. They are, indeed, feasible to select multiple targets. During the selection with the aforementioned methods, all intended targets will be treated independently and selected by loops operations of "hitting down and up" or "aim and confirm". In those cases, there exist pauses among each loop of selection. For hitting virtual targets for selection, especially, the extra time and movement would be required to recover (i.e., from beneath to above the target) after hitting the target. This situation further enlarges the pause among target selections.

Sewing-like interaction attempts to enhance the fluidity of target selections from the following aspects. First, sewing-like interaction uses the immediate confirmation to avoid the time consumption from the confirmation mechanism (e.g., dwell or pressing a button) and the potential mode-switching between navigation and confirmation. Second, the sewing-like interaction regards the penetrated movement when hitting the target as a natural advantage to head for the following target. Compared with bouncing on each virtual target, sewing-like interaction allows users to move directly towards the following target, thus the time and movement for the hitting down recovery will be saved, which improves the selection fluidity.

5.4 SewTyping and Empirical Evaluation

Based on the sewing-like interaction, SewTyping renders the virtual keyboard double-side interactive. Users can use the handheld controller (with the virtual mallet on the top) to perform sewing-like movements among character keys. For instance, users enter the word "ACT" (shown in Figure 5.1) by penetrating down through "A", up through "C", down through "T". For words with repetitive characters (e.g., "WOOD"), users can penetrate through "W", up through "O", down through "O", and up through "D".

The goal of this experiment was to evaluate whether SewTyping can improve VR text entry efficiency. Therefore, we conducted a between-subjects experiment with 18 participants comparing three techniques: Controller Pointing, Drum-like Keyboard, and SewTyping.

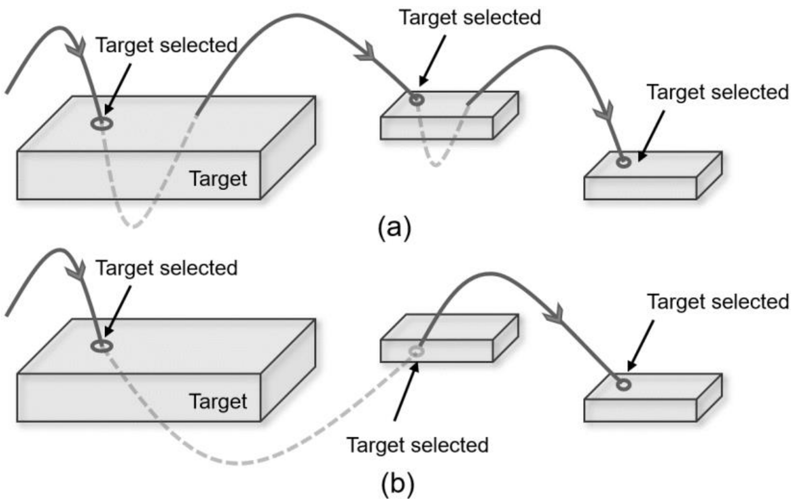


Figure 5.2 Two successive target selection methods: (a) up-and-down "pressing" and (b) sewing-like interaction. Sewing-like interaction leverages the penetrable feature in VR to achieve the successive target selection by penetrating targets as sewing with a needle on the fabric.

5.5.1 *Participants and Apparatus*

We recruited 18 participants for the experiment (6 males and 12 females, aged between 22 and 46 years, *M* = 26.56, *SD* = 5.55, all right-handed, none of them being native English speakers). All participants had normal vision without color blindness (14 of them wore glasses for short sight correction). None of them had VR text entry experience before the experiment. All participants were able to comprehend English sentences and were familiar with the QWERTY keyboard according to their self-report.

The experiment system was implemented based on the Cutie keys (Cutiekeys, 2017) in Unity 5.6 with HTC Vive and its handheld controllers. We set up a spatial area of 2.5m (length) × 2.5m (width) × 2m (height) tracking space with two HTC Vive optical trackers. We used a virtual keyboard following the QWERTY layout. We also ran a pilot study and confirmed the key size 10cm × 10cm, a 3cm gap between keys, 32cm length for the virtual mallet, 1.5cm radius of the sphere on the top of the virtual mallet forming a balance between the detection quality of the system and the participants' typing experience. The system ran on a desktop computer with i7-3770 CPU, 16 GB RAM, NVIDIA Quadro K4000 graphics card, and Windows 10 operating system. Intelligent-aid techniques such as auto-correction and auto-implementation functions were disabled to avoid the distraction of incorrect or unexpected auto-entry conditions.

5.5.2 *Task and Corpus*

In the experiment, all participants were requested to transcribe sentences in the experiment system with the assigned technique as fast and accurate as possible. If any typo was detected during the text entry process, participants had to correct it with the backspace on the virtual keyboard.

We used a subset of the Enron Mobile Email Dataset (Vertanen & Kristensson, 2011) (107 sentences, with 20-40 characters each, all in lowercase without any punctuations or numbers) as the corpus because 1) all sentences are used on a daily basis and they are easy to memorize (Kristensson & Vertanen, 2012) and 2) it shows its validity for evaluating text entry techniques (Speicher et al., 2018).

5.5.3 *Design and Procedure*

According to the literature, we choose Controller Pointing and Drum-like Keyboard as the comparative techniques for VR text entry techniques (especially with virtual keyboards and handheld controllers) because of their promising text entry rates and error rates, as reported in (Speicher et al., 2018) and (Boletsis & Kongsvik, 2019). To be consistent with the evaluation designs in (Speicher et al., 2018) and (Boletsis & Kongsvik, 2019), we asked participants to finish the text entry in the two-hand condition. Here we describe the operational procedures of the three evaluated VR text entry techniques:

- *Controller Pointing*: participants navigate the ray emitted from the handheld controller to point at the intended key and confirm the selection by pressing the trigger button on the controller.

- *Drum-like Keyboard*: participants use the virtual mallet extended from the head of the handheld controller to select the character by hitting the character button from the top side of it.

- *SewTyping*: participants use the virtual mallet extended from the head of the handheld controller to penetrate down or up through buttons to select the intended characters.

The ray emitted from the handheld controller (for Controller Pointing) and the virtual mallet (for Drum-like Keyboard and SewTyping) were visible during the experiment. When the button was irradiated by the ray or hit by the virtual mallet, it would be highlighted in light red. Haptic feedback was provided as the participant pressed the trigger of the handheld controller (for Controller Pointing) or as the button was hit (for Drum-like Keyboard and SewTyping).

To avoid potential interference from the various operation methods of the three techniques (especially for Drum-like Keyboard and SewTyping) on data quality, we chose a between-subjects design in the experiment. The independent variable was the *technique*. We randomly and evenly arranged 18 participants into three groups. For each group, participants were instructed with only one text entry technique and they were requested to transcribe 50 sentences (in five sessions, similar to the design of (C. Yu et al., 2017)) randomly chosen from the corpus. In total, the design was: 3 techniques × 6 participants per technique × 50 sentences = 900 transcriptions.

We first informed participants about the purpose of this study. After participants signed the informed consent, we gave a tutorial on the text entry technique assigned to them (due to the between-subjects design). Then we instructed participants on how to wear the head-mounted display, hold controllers with their left and right hands and sit on the chair in the experiment area where their movement could be detected by the HTC Vive optical trackers (see Figure 5.3). Participants had enough time (around 7-10 minutes) to practice the assigned technique before the formal experiment. During the practice, we also helped participants to adjust the height and orientation of the virtual keyboard and the distance between participants and the virtual keyboard according to participants' feedback.

The target sentence first appeared on the experiment interface. Then participants were required to transcribe the sentence and submit the input by pressing the button on the handheld controller. Participants could have a 2-minute rest after every 10 sentences.

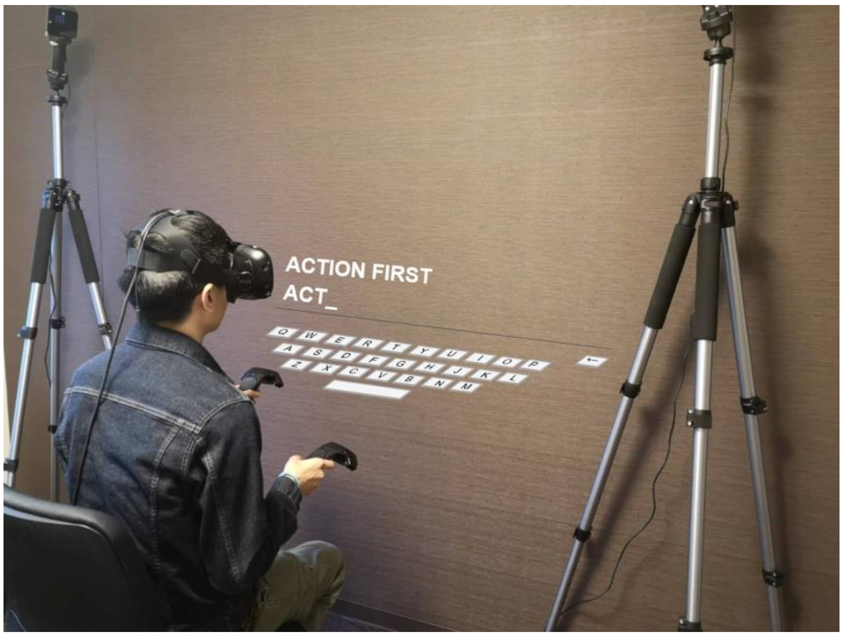


Figure 5.3 A participant in the VR text entry task with handheld controllers and the virtual keyboard floating in the head-mounted display. In this picture, there was an angle of 30 degrees between the virtual keyboard and the ground.

After the experiment, we used the questionnaire deployed in (C. Yu et al., 2017) (five-scale, one for bad, five for good, see Appendix 4) to collect participants' perceived ratings on speed, accuracy, fluidity, fatigue, learnability, and preference. System Usability Scale (SUS) (Brooke, 2013) was also used to collect participants' subjective feedback and comments.

5.5.4 Results

We filtered the data with 3 times the standard deviation of the backspace number and removed 37 (4.11%) outliers. As the data didn't pass the normality check, we analyzed the data with the Kruskal-Wallis test and the post-hoc test using Mann-Whitney tests with Bonferroni correction.

Word per Minute (WPM). For every transcribed sentence, the typing speed is calculated as the number of words (every five characters) divided by the transcription time (in minutes) (Feit et al., 2016). Overall, SewTyping got the highest WPM (*M* = 25.94, *SD* = 6.02). Drum-like Keyboard got a higher WPM (*M* = 22.88, *SD* = 4.38) than Controller Pointing (*M* = 18.22, *SD* = 4.55). A Kruskal-Wallis test revealed a significant effect of technique on the typing speed ($\chi^2(2) = 239.19$, $p < .001$). The post-hoc testing using Mann-Whitney tests with Bonferroni correction showed significant differences among the three techniques (all $p<.001$). Figure 5.4 shows the typing speed of the three techniques among five sessions.

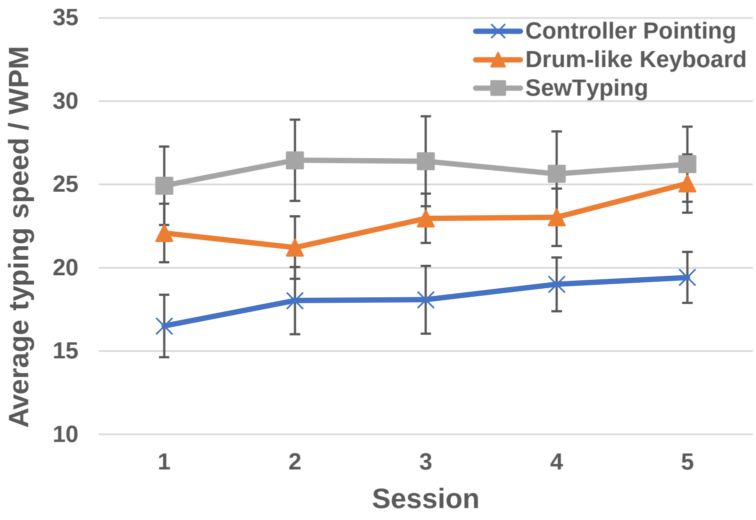


Figure 5.4 The average typing speed for three techniques in five sessions. Error bars represent the standard error.

Inter-key Interval (IKI). IKI is the duration (in milliseconds) between two button selections (Terzuolo & Viviani, 1980) (including space and backspace key). We used IKI to evaluate the fluidity of text entry. SewTyping got the shortest IKI (*M* = 440.48, *SD* = 89.86). Drum-like Keyboard got less IKI (*M* = 493.51, *SD* = 72.88) than Controller Pointing (*M* = 621.37, *SD* = 143.96). A Kruskal-Wallis test revealed a significant effect of technique on the IKI ($\chi^2(2) = 317.81$, $p < .001$). Post-hoc testing using Mann-Whitney tests with Bonferroni correction showed significant differences among the three techniques (all $p<.001$). Figure 5.5 shows the IKI of three techniques among five sessions.

Total Error Rate. As the experiment forced participants to correct typos before submitting the transcription, thus we only evaluated the total error rate (equal to the corrected error rate). Total error rate is calculated as the number of incorrect-but-corrected characters divided by the sum of correct and incorrect-but-corrected characters (Soukoreff & MacKenzie, 2003). The order of the average total error rate for the three techniques was: Controller Pointing (4.86%) > SewTyping (4.19%) > Drum-like Keyboard (3.9%). A Kruskal-Wallis test did not find a significant difference of technique on the total error rate ($p = 0.31$).

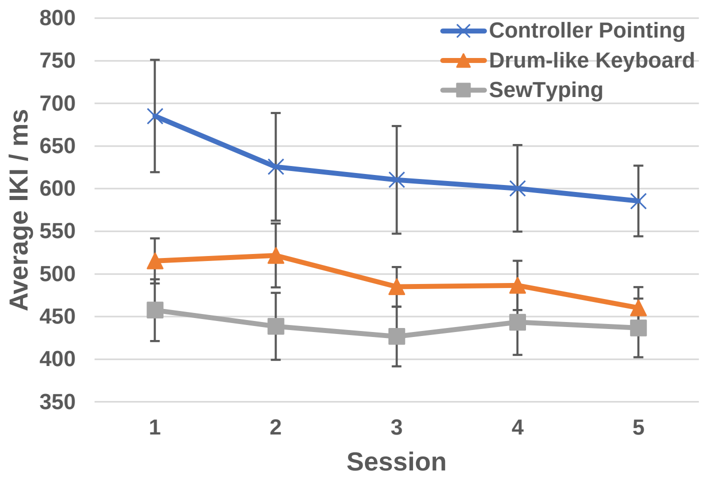


Figure 5.5 The average IKI for three techniques in five sessions. Error bars represent the standard error.

Subjective Evaluations. SUS results showed that Controller Pointing got the lowest score (*M* = 69.6, *SD* = 10.66, calculated by the SUS protocol (Brooke, 2013)), Drum-like Keyboard with an average of 77.5 (*SD* = 14.05), SewTyping with an average of 77.9 (*SD* = 6.79). All three techniques got high ratings (the higher, the better) on question No. 7: “I would imagine that most people would learn to use this system very quickly”).

Evaluations according to the participants’ subjective perceptions were obtained via a 5-scale questionnaire (see Figure 5.6). Results showed that SewTyping got the highest scores on speed (*M* = 4.67, *SD* = 0.52), fluidity (*M* = 4.5, *SD* = 0.55), and learnability (*M* = 4.83, *SD* = 0.41). A Kruskal-Wallis test revealed a significant effect of technique on the speed ($\chi^2(2) = 6.09$, $p < .05$). A post-hoc testing using Mann-Whitney tests with Bonferroni correction revealed the significant differences between Controller Pointing and SewTyping ($p < .05$, $r = 0.58$).

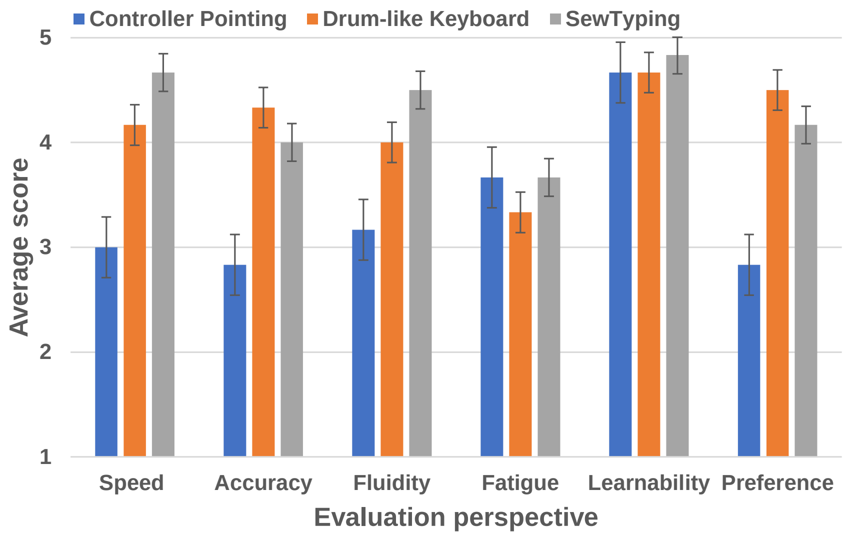


Figure 5.6 Participants' feedback regarding six evaluation perspectives (1 for bad, 5 for good). Error bars represent the standard error.

5.5 Discussion

We facilitated VR text entry performance with SewTyping based on sewing-like interaction. In this section, we discuss the design of sewing-like interaction and interesting findings we got from the empirical study.

From the interaction perspective, sewing-like interaction shares similarities with crossing-based (Accot & Zhai, 2002) target selection in VR (Tu et al., 2019) that:

- They both perform a movement starting from one side to another side of the object.
- Both of them consider the continuous movement as a solution for successive selections.

Whereas, there are also differences between these two methods:

- Sewing-like interaction used a virtual mallet (as a needle) to stimulate the sewing movement (i.e., penetrating the surface of a 3D virtual object); crossing is the operation that behaves more like cutting an object with a beam (i.e., crossing the boundary of the object).
- Sewing-like interaction focuses on the interaction depth from the target perspective (i.e., the thickness of the target), while crossing focuses more on the depth from the VR environment perspective (i.e., the distance between targets and users).
- Sewing-like interaction focuses more on the fluidity enhancement when selecting multiple targets while crossing focuses more on the target selection method itself (crossing-based vs. ray-based).

- Sewing-like interaction allows virtual objects not only interactive from only one side, which provides more flexibilities when managing the penetration path (i.e., not mandatory to execute penetrations from the same side). By considering the penetration depth, various interaction possibilities can be possible when penetrating virtual objects (e.g., rebounding after half-penetration means the selection revoke).

Compared with Controller Pointing and Drum-like Keyboard, SewTyping got the shortest IKI, which implies that SewTyping could reduce the duration of key selections and enhance the fluidity of text entry. The reason is that, because of the participants' familiarity with the keyboard layout, we could leverage the momentum of the penetration movement and save the time and movement by allowing them to move directly through to the next key for subsequent selections (based on the double-side interactive keyboard). Controller Pointing showed longer IKI for character selection because it requires time to complete the confirmation (by pressing the trigger button), release the trigger button, and search for the next key.

From the design perspective, SewTyping avoids the repetitive up-and-down character selection movement (with sudden acceleration changes), which may cause neuromuscular fatigue (Gates & Dingwell, 2008). Although Drum-like Keyboard got increasing typing speed as the session proceeded, participants also encountered extra fatigue during the speed-up. One participant using Drum-like Keyboard commented that: "*I usually strike down keys quickly and hard, thus it brings more fatigue (and it consumes more effort) to stop the strike and start the next one, especially when I attempted to improve my typing speed*". During the practice, it was interesting to observe that participants developed a similar strategy to mitigate the fatigue brought from the technique assigned to the participant. All participants chose to keep elbows close to the bodies during text entry. This strategy reduced the chance of moving the whole arm when using the assigned technique. Similarly, all participants chose to adjust the virtual keyboard in front of them, especially in the area where they could mainly move their forearms instead of the whole arm.

The high learnability rating inferred that SewTyping is easy for participants to use. There are two reasons for this. First, SewTyping maintains the QWERTY layout, which mitigates the burden of learning a new keyboard. Second, the familiarity inspired by the sewing metaphor assists participants to understand the way to interact and adapt to the new technique. Additionally, the sewing metaphor changes the VR text entry from a task to engaging virtual sewing gameplay with needles (handheld controllers) on the fabric (intangible virtual keyboard).

SewTyping enters text with a flexible input unit. In detail, SewTyping can arrange characters (with any length), even the text revision process (i.e., the use of backspace) into a single and fluid movement. Compared with character-level techniques (e.g., Controller Pointing and Drum-like Keyboard), SewTyping enhances the fluidity of text entry with fewer pauses between characters. As for word-level techniques (e.g., GestureType from (C. Yu et al., 2017)), SewTyping allows revision while participants are observing typos or improper content (e.g., misused word) rather than after performing a word-level gesture. It is surprising that SewTyping achieved a higher typing speed (25.94 WPM) than GestureType (24.73 WPM, as reported in (C. Yu et al., 2017)) without the risk of dizziness caused by the frequent head-shaking movement. When typing with two hands, SewTyping also enables the sewing-like movement to shift between hands. This means that participants can avoid long-distance stretching (e.g., from “z” to “o”) and thus improve the typing speed and fluidity.

In the design of VR text entry techniques, SewTyping provides an interesting perspective to consider the influence of the penetrable feature of intangible displays. Instead of merely tolerating the negative effects of the penetration movement, we exploit it as a potential and turn it into an advantage by producing fluid text entry performance.

5.6 Conclusion

In this chapter, we present and validate SewTyping, a novel technique for facilitating efficient VR text entry performance on virtual keyboards with handheld controllers. We take advantage of the penetrable feature of intangible displays to realize the sewing-like interaction. Users can enter multiple characters successively by penetrating up through and down through among keys on the virtual keyboard with the handheld controller. We illustrate the design of SewTyping and validate it with a user study. Results show that participants can quickly get used to SewTyping and reach 25.94 WPM with essential training. Participants regarded SewTyping as easy to learn. Furthermore, SewTyping is easy to implement in VR applications since the keyboard layout remains unchanged. This also avoids extra cognitive load caused by learning and adapting to a new layout.

CHAPTER 6

GENERAL DISCUSSION

6.1 Replacement-based text revision

Swap simplifies the caret control when redesigning the text revision interaction on mobile devices to focus on the revision task itself. This produces the following benefits to text revision efficiency. First, Swap allows participants to enter content for revision as they observe the revision target (with entry speed similar to that of regular text input). This is less disruptive to the flow of mind and input because participants can quickly turn their revision intention into real actions without breaking their focus in order to navigate the caret. Second, Swap turns high-precision caret control between characters into target selection with the expanded layout, which reduces the accuracy requirements of the caret control.

Swap provides a new perspective ("replacement") to interpret the text revision task and to unify various text revision conditions into "type first and then replace". With the symbolized backspace and the expanded layout, Swap visualizes all contents (words, spaces between words, and the symbolized backspace) during the revision and making the replacement intuitive and easy to understand by participants.

Swap improves the deletion efficiency with the symbolized backspace. This diminishes the time consumption and the number of repetitive backspace keystrokes when deleting multiple characters (e.g., correcting Wi errors). In addition, with the symbolized backspace, the conventional backspace could also reveal advantages in quick corrections (e.g. correcting the typo near the caret (Komninos et al., 2018)).

Two user studies showed that Swap can handle both light (e.g., revision in one sentence) and heavy-load (e.g., revising a paragraph) revision conditions. It indicated that Swap could handle most daily text-related conditions, such as instant messaging and long text composition (e.g., email, online notes, blogs, etc.). The concept of replacement and the symbolized backspace also shed light on symbolizing (or visualizing) more commands for various text revision requirements (e.g., bold/italic/underline words, change font size/color, etc.).

To gather more user feedback after studies, we removed most control settings in the user study and asked participants to revise a paragraph with improved Swap. All participants commented that improved Swap could enhance efficiency and fluency when revising multiple targets. Five participants showed their expectations about integrating improved Swap with their own mobile devices such as iPads and laptops with multi-touch screens (e.g., Microsoft Surface).

6.2 Design space exploration for VR text revision

Findings of the study on design space exploration for VR text revision not only provide a basic understanding on how to use backspace and caret for VR text revision but also show considerations and suggestions for the design of future VR text revision tools and techniques. Therefore, here we present a set of general design guidelines for text revision in VR environments, highlighting the findings and discussions mentioned regarding the design space exploration process and experimental results. It should be mentioned again that, although text revision is one of the subtasks in VR text input, it is vital for practitioners to realize the importance of text revision when designing novel VR text input techniques for practical use. Below are the general guidelines:

1) When designing VR text revision solutions, the following factors should be considered and satisfied with high priorities: ease of use, less physical and mental load, and robustness of handling various revision conditions.

2) Mix-granularity deletion mechanisms should be used to make sure the flexibility and efficiency of both word-level deletion and character-level quick correction. Meanwhile, assistive techniques are necessary to enhance the exception handling capacities during revision. For instance, providing proper visual feedback to indicate the word-level changes

or using recovery functions such as undo and prediction list to minimize the cost of mis-deletion.

3) Caret control mechanisms should be involved in VR text revision design to provide the flexibility of controlling the caret or other similar widgets. Position correction algorithms should also be considered for better caret control smoothness and accuracy.

4) As backspace and caret control can be frequently used in conditions with heavy revision loads (e.g., editing articles), these two should be implemented with less and subtle movement to avoid physical fatigue (e.g., controller button pressing vs. striking the backspace with fore-arm movement).

6.3 Sewing-like interaction and SewTyping

Our evaluation validated the feasibility of the sewing-like interaction in the context of VR text entry. Besides that, we also found great potentials to apply the sewing-like interaction in more practical scenarios:

Spatial gesture design for gaming: Sewing-like interaction sheds light on extending input channels (e.g., the spatial orientation or the amplitude when performing the gesture) for current gesture designs. For instance, in a VR wizard game, users can perform a vertical “N” gesture (with the handheld controller) for the magic spell with normal power. They can also perform a horizontal larger “N” gesture for the same spell, but with enhanced power to strike the enemy.

Spatial menu design: 1) We can implement a spatial multi-layer menu and use the sewing-like interaction to realize the fluid process of calling out the menu, selecting the category, and selecting the intended item under the selected category with a single controller movement. 2) We can also arrange various functions into categories and integrate them into only one virtual button. Users can trigger different categories by penetrating the button at different angles and then penetrate the intended item for selection.

Non-visual text entry: Similar to (Yi et al., 2015), with trajectory detection and machine learning algorithms, sewing-like interaction and users’ familiarity with the QWERTY layout can be leveraged to achieve freehand text entry in VR without showing the virtual keyboard.

Surgical training: As users can use the handheld controller to simulate the real-life sewing behavior (penetration movement) in VR environments, a series of applications can be implemented based on the sewing-like interaction to imitate medical training such as surgical wound closure and venipuncture.

CHAPTER 7

LIMITATION AND FUTURE WORK

For Swap, our current study mainly focused on the tool design and interaction design for text revision on mobile devices. Therefore, it encountered several limitations which inspire us to perform further evaluations and development. First, quantitative results and subjective evaluations validated the feasibility and the advantages of the replacement-based text revision techniques, whereas a long-term in-the-wild study is still needed to investigate text revision efficiency and changes in user behaviors and how to facilitate them efficiently.

Second, to make the replacement-based text revision technique compatible with real-life scenarios, more factors could be considered in the following research, such as mobility (e.g., walking), workload (e.g., one-hand occupied), keyboard layout, and the size of the touchscreen.

Moreover, it should be noted that multiple intelligent text entry aid techniques and algorithms have been embedded into current text input methods. These can also be integrated with Swap for quick and seamless text revision operations.

For the text revision exploration in VR, our current work mainly focuses on the design space exploration of VR text revision based on backspace and caret. However, it should be noted that, in practical use, text revision is often mixed with regular text entry processes, and users often need to revise multiple targets. Therefore, one of the future work is to conduct long-term research to investigate further on how the proposed techniques could help users to enhance VR typing (and text revision) performance in real-world situations. Due to the variety of input devices (e.g., stylus or joysticks) applied in VR systems, further implementations and evaluations will be scheduled to determine which device is more capable of handling real-life VR text-related tasks. Additionally, further studies will also examine the influence of different caret shapes and various conditions (e.g., sitting or standing) on users' VR text revision performance.

For SewTyping, there are the following aspects for future work. First, as we didn't evaluate SewTyping in the one-hand condition, further study will be conducted to investigate typing performance of SewTyping under both unimanual and bimanual conditions. Second, as some text entry techniques may include intelligent aids (e.g., auto-correction), we will investigate the influence of those techniques on SewTyping performance in a future study. Third, to further understand the sewing-like interaction, a series of studies will be conducted to evaluate the influence of factors (e.g., the orientation of the target, target size, target shape, the gap between targets, visual and haptic feedback, length of the virtual mallet, etc.) on users' interaction performance. Last but not least, with the popularity of auto-aid techniques (e.g., autocorrection algorithms), we will also integrate those smart features into the design of SewTyping and make it as a practical and powerful tool to facilitate users' VR typing performance.

CHAPTER 8

CONCLUSION

In the foreseeable future, text input will still be essential and vital when users interacting with mobile devices and VR systems. In that caser, effective and well-designed (according to the scenario) text input methods could bring more benefits and satisfaction to users when they type. Our target is to re-design the mobile and VR text entry and text revision tools and processes to make them more adaptive to the applied scenarios.

Instead of using the existed text input solutions (that applied on physical keyboards), we present two improvement designs. First, we propose a novel interaction paradigm (Swap) and related techniques for effective text revision with less use of backspace and caret control. Compared with the conventional text revision process (based on the caret and backspace), Swap uses the replacement operation to unify the text revision process regarding various text revision conditions. By allowing users to enter the revision content first, Swap can either leverage users' regular typing speed during text revision or decrease the potential of committing cascading mis-operations. We validate the feasibility of Swap by conducting a series of comparative user studies between Swap-based and conventional text revision techniques. Apart from improving users' text revision performance, we also combine the daily-life sewing metaphor with the penetrable feature of intangible displays to enhance text entry efficiency and fluidity in VR systems.

Our work provides new perspectives to criticize current text input solutions applied in mobile devices and to seek opportunities to improve users' text input satisfaction and efficiency. We hope that, through our attempts, users could focus on the real tasks rather than text input itself when typing on mobile devices and VR applications.

APPENDIX 1

NASA-TLX

NASA-TLX is a subjective assessment tool to measure users' workload in a certain task from the following perspectives: mental demand, physical demand, temporal demand, performance, effort, and frustration. Details are illustrated in the webpage from NASA (https://humansystems.arc.nasa.gov/groups/TLX/).

Instruction. After the user finished the task, the user will receive the NASA-TLX questionnaire. Then the user needs to recall and self-evaluate his/her performance in the task. Finally, the user will be required to mark their load on the 21-scale axis for each of five perspectives.

Physical Demand	How mentally demanding was the task?

Very low Very high

Physical Demand	How physically demanding was the task?

Very low Very high

Temporal Demand	How hurried or rushed was the pace of the task?

Very low Very high

Performance How successful were you in accomplishing what you were asked to do?

Very low Very high

Effort How hard did you have to work to accomplish your level of performance?

Very low Very high

Frustration How insecure, discouraged, irritated, stressed, and annoyed were you?

Very low Very high

APPENDIX 2

7-POINT LIKERT-SCALE FOR SWAP

This 7-point Likert-scale is used to evaluate users' subject attitudes towards our proposed text revision technique on mobile devices. We ask the user to rank on the complexity, fatigue, difficulty, dislike, and frustration when using the technique (1 for the lowest, 7 for the highest).

Perspective							
Complexity	1	2	3	4	5	6	7
Fatigue	1	2	3	4	5	6	7
Difficulty	1	2	3	4	5	6	7
Dislike	1	2	3	4	5	6	7
Frustration	1	2	3	4	5	6	7

APPENDIX 3

SYSTEM USABILITY SCALE (SUS)

Here list the 10 questions in the SUS we used for gathering users' feedback. Users were requested to give scores on those 10 questions with the scale of 1-5 (1 for strongly disagree, 5 for strongly agree).

Questions:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome/awkward to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

APPENDIX 4

5-POINT LIKERT-SCALE FOR SEWTYPING

This 5-point Likert-scale is used to evaluate users' subject attitudes towards our proposed VR text entry technique. We ask the user to rank on the speed, accuracy, fluidity, fatigue, learnability, and preference when using the assigned technique (1 for bad, 5 for good).

Perspective					
Speed	1	2	3	4	5
Accuracy	1	2	3	4	5
Fluidity	1	2	3	4	5
Fatigue	1	2	3	4	5
Learnability	1	2	3	4	5
Preference	1	2	3	4	5

BIBLIOGRAPHY

360marketupdates. (2019). *2019-2024 Global and Regional Augmented Reality and Virtual Reality Industry Production, Sales and Consumption Status and Prospects Professional Market Research Report*. https://www.360marketupdates.com/2019-2024-global-and-regional-augmented-reality-and-virtual-reality-industry-production-sales-and-consumption-status-and-prospects-professional-market-research-report-13733991

Accot, J., & Zhai, S. (2002). More than dotting the i's- Foundations for crossing-based interfaces. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/503376.503390

Alharbi, O., Arif, A. S., Stuerzlinger, W., Dunlop, M. D., & Komninos, A. (2019). WiseType: A Tablet Keyboard with Color-Coded Visualization and Various Editing Options for Error Correction. *Proceedings of Graphics Interface 2019*. https://doi.org/10.20380/GI2019.04

Ali, Z. (2017). *Increase Text Delete Speed On Stock iOS Keyboard With 3D Touch*. https://ioshacker.com/how-to/adjust-text-delete-speed-stock-ios-keyboard-3d-touch

Ando, T., Isomoto, T., Shizuki, B., & Takahashi, S. (2019). One-handed Rapid Text Selection and Command Execution Method for Smartphones. *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, LBW0224:1--LBW0224:6. https://doi.org/10.1145/3290607.3312850

Ando, T., Isomoto, T., Shizuki, B., & Takahashi, S. (2018). Press &#38; Tilt: One-handed Text Selection and Command Execution on Smartphone. *Proceedings of the 30th Australian Conference on Computer-Human Interaction*, 401–405. https://doi.org/10.1145/3292147.3292178

Android. (2017). *Spell checker framework*. https://developer.android.com/guide/topics/text/spell-checker-framework

Apple. (2019). *Caret navigation features in iOS 13*. https://www.apple.com/ios/ios-13-preview/features

Arbesman, S. (2017). *Overcomplicated: Technology at the Limits of Comprehension*. Portfolio. https://www.amazon.com/Overcomplicated-Technology-at-Limits-Comprehension/dp/0143131303?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0143131303

Arif, A. S. (2015). *Predicting and Reducing the Impact of Errors in Character-Based Text Entry*. York University.

Arif, A. S., Kim, S., Stuerzlinger, W., Lee, G., & Mazalek, A. (2016). Evaluation of a Smart-Restorable Backspace Technique to Facilitate Text Entry Error Correction. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 5151–5162. https://doi.org/10.1145/2858036.2858407

Arif, A. S., Pahud, M., Hinckley, K., & Buxton, B. (2014). Experimental Study of Stroke Shortcuts for a Touchscreen Keyboard with Gesture-redundant Keys Removed. *Proceedings of Graphics Interface 2014*, 43–50. http://dl.acm.org/citation.cfm?id=2619648.2619657

Arif, A. S., & Stuerzlinger, W. (2010). Predicting the Cost of Error Correction in Character-based Text Entry Technologies. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 5–14. https://doi.org/10.1145/1753326.1753329

Arif, A. S., & Stuerzlinger, W. (2013). Pseudo-pressure Detection and Its Use in Predictive Text Entry on Touchscreens. *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, 383–392. https://doi.org/10.1145/2541016.2541024

Arnold, K. C., Gajos, K. Z., & Kalai, A. T. (2016). On Suggesting Phrases vs. Predicting Words for Mobile Text Composition. *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, 603–608. https://doi.org/10.1145/2984511.2984584

Boletsis, C., & Kongsvik, S. (2019). Controller-based Text-input Techniques for Virtual Reality: An Empirical Comparison. *International Journal of Virtual Reality*. https://doi.org/10.20870/ijvr.2019.19.3.2917

Boustila, S., Guegan, T., Takashima, K., & Kitamura, Y. (2019). Text typing in VR using smartphones touchscreen and HMD. *26th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019 - Proceedings*. https://doi.org/10.1109/VR.2019.8798238

Bovet, S., Kehoe, A., Crowley, K., Curran, N., Gutierrez, M., Meisser, M., Sullivan, D. O., & Rouvinez, T. (2018). Using Traditional Keyboards in VR: SteamVR Developer Kit and Pilot Game User Study. *2018 IEEE Games, Entertainment, Media Conference, GEM 2018*. https://doi.org/10.1109/GEM.2018.8516449

Bowman, D. A, Ly, V. Q., & Campbell, J. M. (2001). Pinch keyboard: Natural text input for immersive virtual environments. *Virginia Tech Dept. of Computer Science, Technical Report*.

Bowman, Doug A., Rhoton, C. J., & Pinho, M. S. (2002). Text Input Techniques for Immersive Virtual Environments: An Empirical Comparison. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. https://doi.org/10.1177/154193120204602611

Brewster, S. (2002). Overcoming the Lack of Screen Space on Mobile Computers. *Personal Ubiquitous Comput.*, *6*(3), 188–205. https://doi.org/10.1007/s007790200019

Brooke, J. (2013). SUS: a retrospective. *Journal of Usability Studies*.

Brun, D., George, S., & Gouin-Vallerand, C. (2019). Keycube is a kind of keyboard (K3). *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290607.3313258

Buschek, D., Bisinger, B., & Alt, F. (2018). ResearchIME: A mobile keyboard application for studying free typing behaviour in the wild. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3173574.3173829

Card, S. K., Newell, A., & Moran, T. P. (1983). *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc.

Chan, L. W., Kao, H. S., Chen, M. Y., Lee, M. S., Hsu, J., & Hung, Y. P. (2010). Touching the void: Direct-touch interaction for intangible displays. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/1753326.1753725

Chen, S., Wang, J., Guerra, S., Mittal, N., & Prakkamakul, S. (2019). Exploring word-gesture text entry techniques in virtual reality. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290607.3312762

Choi, D., Cho, H., Seo, K., Lee, S., Lee, J., & Ko, J. (2019). Designing Hand Pose Aware Virtual Keyboard with Hand Drift Tolerance. *IEEE Access*. https://doi.org/10.1109/ACCESS.2019.2929310

Colle, H. A., & Hiszem, K. J. (2004). Standing at a kiosk: Effects of key size and spacing on touch screen numeric keypad performance and user preference. *Ergonomics*, *47*(13), 1406–1423. https://doi.org/10.1080/00140130410001724228

Cutiekeys. (2017). *NormalVR/CutieKeys*. https://github.com/NormalVR/CutieKeys/

Cutts, M. (2013). *Oxford Guide to Plain English (Oxford Paperback Reference)*. OUP Oxford. https://www.xarg.org/ref/a/B00FZSX76G/

Dash, S. (2017). BlueTap - The Ultimate Virtual-Reality (VR) Keyboard. In *Medium*. Eunoia.i/o. https://medium.com/eunoia-i-o/bluetap-the-ultimate-virtual-reality-vr-keyboard-77f1e3d57d6f

Dhakal, V., Feit, A. M., Kristensson, P. O., & Oulasvirta, A. (2018). Observations on Typing from 136 Million Keystrokes. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 646:1--646:12. https://doi.org/10.1145/3173574.3174220

Dube, T. J., & Arif, A. S. (2019). Text Entry in Virtual Reality: A Comprehensive Review of the Literature. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-030-22643-5_33

Dudley, J., Benko, H., Wigdor, D., & Kristensson, P. O. (2019). Performance envelopes of virtual keyboard text input strategies in virtual reality. *Proceedings - 2019 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2019*. https://doi.org/10.1109/ISMAR.2019.00027

Dudley, J. J., Vertanen, K., & Ola Kristensson, P. (2018). Fast and precise touch-based text entry for head-mounted augmented reality with variable occlusion. *ACM Transactions on Computer-Human Interaction*. https://doi.org/10.1145/3232163

Eady, A. K., & Girouard, A. (2015). Caret Manipulation Using Deformable Input in Mobile Devices. *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, 587–591. https://doi.org/10.1145/2677199.2687916

Fashimpaur, J., Kin, K., & Longest, M. (2020). *PinchType: Text Entry for Virtual and Augmented Reality Using Comfortable Thumb to Fingertip Pinches*. https://doi.org/10.1145/3334480.3382888

Feit, A. M., Weir, D., & Oulasvirta, A. (2016). How we type: Movement strategies and performance in everyday typing. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/2858036.2858233

Fleksy. (2019). *Fleksy keyboard*. https://www.fleksy.com

Fuccella, V., Isokoski, P., & Martin, B. (2013). Gestures and Widgets: Performance in Text Editing on Multi-touch Capable Mobile Devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2785–2794. https://doi.org/10.1145/2470654.2481385

Gates, D. H., & Dingwell, J. B. (2008). The effects of neuromuscular fatigue on task performance during repetitive goal-directed movements. *Experimental Brain Research*. https://doi.org/10.1007/s00221-008-1326-8

Gentner, D. R., Grudin, J. T., Larochelle, S., Norman, D. A., & Rumelhart, D. E. (1983). A Glossary of Terms Including a Classification of Typing Errors. In *Cognitive Aspects of Skilled Typewriting*. https://doi.org/10.1007/978-1-4612-5470-6_2

González, G., Molina, J. P., García, A. S., Martínez, D., & González, P. (2009). Evaluation of text input techniques in immersive virtual environments. In *New Trends on Human-Computer Interaction: Research, Development, New Tools and Methods*. https://doi.org/10.1007/978-1-84882-352-5_11

Google. (2018a). *Magnifier widget*. https://developer.android.com/guide/topics/text/magnifier

Google. (2018b). *Selection handle for Android Textview*. https://developer.android.com/reference/android/widget/TextView.html#getTextSelectHandle()

Google. (2019). *Gboard*. https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin

GoogleDaydream. (2016). *Daydream Labs: exploring and sharing VR's possibilities*. https://blog.google/products/daydream/daydream-labs-exploring-and-sharing-vrs

GooglePatent. (2017). *Input device interaction-US20170090747A1*. https://patents.google.com/patent/US20170090747A1/en?oq=20170090747

Grammarly. (2019). *Grammarly: free writing assistant*. https://www.grammarly.com

Grubert, J., Ofek, E., Pahud, M., & Kristensson, P. O. (2018). The Office of the Future: Virtual, Portable, and Global. *IEEE Computer Graphics and Applications*. https://doi.org/10.1109/MCG.2018.2875609

Grubert, J., Witzani, L., Ofek, E., Pahud, M., Kranz, M., & Kristensson, P. O. (2018a). Effects of Hand Representations for Typing in Virtual Reality. *25th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2018 - Proceedings*. https://doi.org/10.1109/VR.2018.8446250

Grubert, J., Witzani, L., Ofek, E., Pahud, M., Kranz, M., & Kristensson, P. O. (2018b). Text Entry in Immersive Head-Mounted Display-Based Virtual Reality Using Standard Keyboards. *25th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2018 - Proceedings*. https://doi.org/10.1109/VR.2018.8446059

Gugenheimer, J., Dobbelstein, D., Winkler, C., Haas, G., & Rukzio, E. (2016). Facetouch: Enabling touch interaction in display fixed uis for mobile virtual reality. *UIST 2016 - Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. https://doi.org/10.1145/2984511.2984576

Guo, J., Weng, D., Zhang, Z., Liu, Y., & Wang, Y. (2019). Evaluation of maslows hierarchy of needs on long-term use of HMDs - A case study of office environment. *26th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019 - Proceedings*. https://doi.org/10.1109/VR.2019.8797972

Gupta, A., Ji, C., Yeo, H. S., Quigley, A., & Vogel, D. (2019). Rotoswype: Word-Gesture Typing using a Ring. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290605.3300244

Hagiya, T., Horiuchi, T., & Yazaki, T. (2016). Typing Tutor: Individualized Tutoring in Text Entry for Older Adults Based on Input Stumble Detection. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, 733–744. https://doi.org/10.1145/2858036.2858455

Hart, S. G. (2006). NASA-task load index (NASA-TLX); 20 years later. *Proceedings of the Human Factors and Ergonomics Society*.

Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. *Advances in Psychology*. https://doi.org/10.1016/S0166-4115(08)62386-9

Hirzle, T., Gugenheimer, J., Geiselhart, F., Bulling, A., & Rukzio, E. (2019). A design space for gaze interaction on head-mounted displays. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290605.3300855

Hoppe, A. H., Otto, L., van de Camp, F., Stiefelhagen, R., & Unmüßig, G. (2018). qVRty: Virtual keyboard with a haptic, real-world representation. *Communications in Computer and Information Science*. https://doi.org/10.1007/978-3-319-92279-9_36

HTC Corporation. (2015). *HTC Vive*. Vive.Com.

idownloadblog. (2018). *How to use iOS 12 virtual keyboard in trackpad mode on iPhones without 3D Touch*. https://www.idownloadblog.com/2018/08/22/howto-iphone-keyboard-trackpad-mode/

Ishii, A., Adachi, T., Shima, K., Nakamae, S., Shizuki, B., & Takahashi, S. (2017). FistPointer: Target Selection Technique Using Mid-air Interaction for Mobile VR Environment. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. https://doi.org/10.1145/3027063.3049795

Janzen, T. B., Thompson, W. F., Ammirante, P., & Ranvaud, R. (2014). Timing skills and expertise: Discrete and continuous timed movements among musicians and athletes. *Frontiers in Psychology*. https://doi.org/10.3389/fpsyg.2014.01482

Jiang, H., & Weng, D. (2020). HiPad: Text entry for Head-Mounted Displays Using Circular Touchpad. *Proceedings - 2020 IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2020*. https://doi.org/10.1109/VR46266.2020.1581236395562

Jimenez, J. G., & Schulze, J. P. (2018). Continuous-motion text input in virtual reality. *IS and T International Symposium on Electronic Imaging Science and Technology*. https://doi.org/10.2352/ISSN.2470-1173.2018.03.ERVR-450

Keyboard - History of the Modern Computer Keyboard. (2019). https://history-computer.com/ModernComputer/Basis/keyboard.html

Kim, Y. R., & Kim, G. J. (2017). HoVR-Type: Smartphone as a typing interface in VR using hovering. *2017 IEEE International Conference on Consumer Electronics, ICCE 2017*. https://doi.org/10.1109/ICCE.2017.7889285

Knierim, P., Kosch, T., Groschopp, J., & Schmidt, A. (2020). *Opportunities and Challenges of Text Input in Portable Virtual Reality*. https://doi.org/10.1145/3334480.3382920

Knierim, P., Schwind, V., Feit, A. M., Nieuwenhuizen, F., & Henze, N. (2018). Physical keyboards in Virtual reality: Analysis of typing performance and effects of avatar hands. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3173574.3173919

Komninos, A., Dunlop, M., Katsaris, K., & Garofalakis, J. (2018). A Glimpse of Mobile Text Entry Errors and Corrective Behaviour in the Wild. *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, 221–228. https://doi.org/10.1145/3236112.3236143

Komninos, A., Nicol, E., & Dunlop, M. D. (2015). Designed with Older Adults to SupportBetter Error Correction in SmartPhone Text Entry: The MaxieKeyboard. *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, 797–802. https://doi.org/10.1145/2786567.2793703

Kristensson, P. O., & Vertanen, K. (2012). Performance comparisons of phrase sets and presentation styles for text entry evaluations. *International Conference on Intelligent User Interfaces, Proceedings IUI*. https://doi.org/10.1145/2166966.2166972

Kuester, F., Chen, M., Phair, M. E., & Mehring, C. (2006). Towards keyboard independent touch typing in VR. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST*. https://doi.org/10.1145/1101616.1101635

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady, 10*, 707.

Li, Y., Sarcar, S., Kim, S., & Ren, X. (2020). Swap: A Replacement-based Text Revision Technique for Mobile Devices. *CHI '20*. https://doi.org/10.1145/3313831.3376217

Lin, J. W., Han, P. H., Lee, J. Y., Chen, Y. S., Chang, T. W., Chen, K. W., & Hung, Y. P. (2017). Visualizing the keyboard in virtual reality for enhancing immersive experience. *ACM SIGGRAPH 2017 Posters, SIGGRAPH 2017*. https://doi.org/10.1145/3102163.3102175

Lu, X., Yu, D., Liang, H. N., Feng, X., & Xu, W. (2019). DepthText: Leveraging head movements towards the depth dimension for hands-free text entry in mobile virtual reality systems. *26th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019 - Proceedings*. https://doi.org/10.1109/VR.2019.8797901

Ma, C., Du, Y., Teng, D., Chen, J., Wang, H., & Dai, G. (2009). An adaptive sketching user interface for education system in virtual reality. *ITME2009 - Proceedings 2009 IEEE International Symposium on IT in Medicine and Education*. https://doi.org/10.1109/ITIME.2009.5236314

Ma, X., Yao, Z., Wang, Y., Pei, W., & Chen, H. (2018). Combining brain-computer interface and eye tracking for high-speed text entry in virtual reality. *International Conference on Intelligent User Interfaces, Proceedings IUI*. https://doi.org/10.1145/3172944.3172988

MacKenzie, I. S., & Soukoreff, R. W. (2003). Phrase sets for evaluating text entry techniques. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/765891.765971

Mcgill, M., Boland, D., Murray-Smith, R., & Brewster, S. (2015). A dose of reality: Overcoming usability challenges in VR head-mounted displays. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/2702123.2702382

Mehring, C., Kuester, F., Singh, K. D., & Chen, M. (2004). KITTY: Keyboard independent touch typing in VR. *Proceedings - Virtual Reality Annual International Symposium*. https://doi.org/10.1109/VR.2004.1310090

Menzner, T., Otte, A., Gesslein, T., Grubert, J., Gagel, P., & Schneider, D. (2019). A capacitive-sensing physical keyboard for VR text entry. *26th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019 - Proceedings*. https://doi.org/10.1109/VR.2019.8797754

Microsoft. (2018). *Using an Input Method Editor in a Game*. https://docs.microsoft.com/en-us/windows/win32/dxtecharts/using-an-input-method-editor-in-a-game

Microsoft. (2019). *SwiftKey: The Smart Keyboard*. https://www.microsoft.com/en-us/swiftkey

Min, K. (2011). Text input tool for immersive VR based on 3 x 3 screen cells. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-24082-9_94

Naskar, A. (2019). The history of the invention of the typewriter. In *Online typing*. Online Typing. https://onlinetyping.org/blog/invention-of-the-typewriter.php

Nguyen, C., DiVerdi, S., Hertzmann, A., & Liu, F. (2017). CollaVR: Collaborative in-headset review for VR video. *UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. https://doi.org/10.1145/3126594.3126659

Norman, D. A. (1995). *The Psychopathology of Everyday Things*. Elsevier.

Ogawa, N., Narumi, T., Kuzuoka, H., & Hirose, M. (2020). Do You Feel Like Passing Through Walls?: Effect of Self-Avatar Appearance on Facilitating Realistic Behavior in Virtual Environments. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems,* 1–14. https://doi.org/10.1145/3313831.3376562

Ogitani, T., Arahori, Y., Shinyama, Y., & Gondow, K. (2018). Space saving text input method for head mounted display with virtual 12-key keyboard. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*. https://doi.org/10.1109/AINA.2018.00059

Otte, A., Menzner, T., Gesslein, T., Gagel, P., Schneider, D., & Grubert, J. (2019). Towards utilizing touch-sensitive physical keyboards for text entry in virtual reality. *26th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019 - Proceedings*. https://doi.org/10.1109/VR.2019.8797740

Otte, A., Schneider, D., Menzner, T., Gesslein, T., Gagel, P., & Grubert, J. (2019). Evaluating text entry in virtual reality using a touch-sensitive physical keyboard. *Adjunct Proceedings of the 2019 IEEE International Symposium on Mixed and Augmented Reality, ISMAR-Adjunct 2019*. https://doi.org/10.1109/ISMAR-Adjunct.2019.000-4

Pham, D. M., & Stuerzlinger, W. (2019). HawKEY: Efficient and versatile text entry for virtual reality. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST*. https://doi.org/10.1145/3359996.3364265

Prätorius, M., Burgbacher, U., Valkov, D., & Hinrichs, K. (2015). Sensing Thumb-to-Finger Taps for Symbolic Input in VR/AR Environments. *IEEE Computer Graphics and Applications*. https://doi.org/10.1109/MCG.2015.106

Prätorius, M., Valkov, D., Burgbacher, U., & Hinrichs, K. (2014). DigiTap: An eyes-free VR/AR symbolic input device. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST*. https://doi.org/10.1145/2671015.2671029

Rajanna, V., & Hansen, J. P. (2018). Gaze typing in virtual reality: Impact of keyboard design, selection method, and motion. *Eye Tracking Research and Applications Symposium (ETRA)*. https://doi.org/10.1145/3204493.3204541

Raymond. (2007). *Whose idea was it to make Ctrl+Backspace delete the previous word*. https://devblogs.microsoft.com/oldnewthing/20071011-00/?p=24823

Rizzo, A. (2019). *Virtual reality for psychological and neurocognitive interventions*. Springer.

Ruan, S., Wobbrock, J. O., Liou, K., Ng, A., & Landay, J. A. (2018). Comparing Speech and Keyboard Text Entry for Short Messages in Two Languages on Touchscreen Phones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. https://doi.org/10.1145/3161187

Scheibel, J.-B., Pierson, C., Martin, B., Godard, N., Fuccella, V., & Isokoski, P. (2013). Virtual Stick in Caret Positioning on Touch Screens. *Proceedings of the 25th Conference on L'Interaction Homme-Machine*, 107:107--107:114. https://doi.org/10.1145/2534903.2534918

Siek, K. A., Rogers, Y., & Connelly, K. H. (2005). Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs. *Proceedings of the 2005 IFIP TC13 International Conference on Human-Computer Interaction,* 267–280. https://doi.org/10.1007/11555261_24

Sindhwani, S., Lutteroth, C., & Weber, G. (2019). ReType: Quick Text Editing with Keyboard and Gaze. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 203:1--203:13. https://doi.org/10.1145/3290605.3300433

Son, J., Ahn, S., Kim, S., & Lee, G. (2019). Improving two-thumb touchpad typing in virtual reality. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290607.3312926

Soukoreff, R. W., & MacKenzie, I. S. (2003). Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. *Conference on Human Factors in Computing Systems - Proceedings*.

Speicher, M., Feit, A. M., Ziegler, P., & Krüger, A. (2018). Selection-based text entry In Virtual Reality. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3173574.3174221

Susu, H., Daqing, Q., Jiabin, Y., & Huawei, T. (2019). Review of studies on target acquisition in virtual reality based on the crossing paradigm. *Virtual Reality & Intelligent Hardware*. https://doi.org/10.3724/sp.j.2096-5796.2019.0006

Suzuki, K., Okabe, K., Sakamoto, R., & Sakamoto, D. (2015). Fix and Slide: Caret Navigation with Movable Background. *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, 79–80. https://doi.org/10.1145/2815585.2815728

Terzuolo, C. A., & Viviani, P. (1980). Determinants and characteristics of motor patterns used for typing. *Neuroscience*. https://doi.org/10.1016/0306-4522(80)90188-8

Tu, H., Huang, S., Yuan, J., Ren, X., & Tian, F. (2019). Crossing-based selection with virtual reality head-mounted displays. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290605.3300848

Vertanen, K., & Kristensson, P. O. (2011). A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails. *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, 295–298. https://doi.org/10.1145/2037373.2037418

Walker, J., Li, B., Vertanen, K., & Kuhl, S. (2017). Efficient typing on a visually occluded physical keyboard. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3025453.3025783

Wikipedia. (2019). *Backspace*. https://en.wikipedia.org/wiki/Backspace

Wilson, A. D., & Agrawala, M. (2006). Text entry using a dual joystick game controller. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/1124772.1124844

Wu, C. M., Hsu, C. W., Lee, T. K., & Smith, S. (2017). A virtual reality keyboard with realistic haptic feedback in a fully immersive virtual environment. *Virtual Reality*. https://doi.org/10.1007/s10055-016-0296-6

Xu, W., Liang, H. N., Zhao, Y., Zhang, T., Yu, Di., & Monteiro, Di. (2019). RingText: Dwell-free and hands-free Text Entry for Mobile Head-Mounted Displays using Head Motions. *IEEE Transactions on Visualization and Computer Graphics*. https://doi.org/10.1109/TVCG.2019.2898736

Yanagihara, N., & Shizuki, B. (2018). Cubic keyboard for virtual reality. *SUI 2018 - Proceedings of the Symposium on Spatial User Interaction*. https://doi.org/10.1145/3267782.3274687

Yi, X., Yu, C., Zhang, M., Gao, S., Sun, K., & Shi, Y. (2015). ATK: Enabling ten-finger freehand typing in air based on3D hand tracking data. *UIST 2015 - Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*. https://doi.org/10.1145/2807442.2807504

Yu, C., Gu, Y., Yang, Z., Yi, X., Luo, H., & Shi, Y. (2017). Tap, dwell or gesture?: Exploring head-based text entry techniques for HMDs. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3025453.3025964

Yu, D., Fan, K., Zhang, H., Monteiro, D., Xu, W., & Liang, H. N. (2018). PizzaText: Text entry for virtual reality systems using dual thumbsticks. *IEEE Transactions on Visualization and Computer Graphics*. https://doi.org/10.1109/TVCG.2018.2868581

Zhang, M., Wen, H., & Wobbrock, J. O. (2019). Type, then correct: Intelligent text correction techniques for mobile text entry using neural networks. *UIST 2019 - Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. https://doi.org/10.1145/3332165.3347924

ACKNOWLEDGMENT

Time passes, all memorable moments seem just like yesterday. For the past three years, I would say nothing but appreciations to those who helped me a lot during the journey of getting my doctor degree.

Sincere gratitude goes to the Kochi University of Technology and its Special Scholarship Program (SSP) for supporting both my research activities and living.

Great appreciation goes to my supervisor, Prof. Xiangshi Ren in the Center for Human-Engaged Computing (CHEC) for the insightful suggestions, high-level guidance, and the platform he built for better research opportunities and global collaborations in the field of Human-Computer Interaction.

I would like to thank my co-supervisors Associate Prof. Yoshiaki Takada, Associate Prof. Yokinobu Hoshino and committee members Associate Prof. Toru Kurihara, Associate Prof. Yoshida Shinichi for their considerate and insightful suggestions on my research.

Special thanks and gratitude go to Assistant Professor Sayan Sarcar for his selfless guidance and suggestions. I cannot forget the days and nights we discussed during the CHI submission. He taught me a lot about how to be an experienced researcher. “Never give up.” is the most valuable spirit that I learned from him.

I would thank Prof. Antti Oulasvirta for providing me the chance of visiting Aalto University, Finland. I would also thank Dr. Sunjun Kim for the insightful discussions and Viet Ba Hirvola for the great assistance on my projects during the research period in Finland.

Special thanks to my 10-year friend, Xinhui Jiang. It is precious for me to have such a friend with such a long time discussing, working, and fighting for the future together.

I would like to thank Kyoko Hatakenaka for her considerate assistance in CHEC. With her selfless help, we could put 100% of our power and passion into the HCI research field. Great appreciations also go to John Cahill for his expert-level proofreading and comments on my work during paper submissions. I would express my thanks to all staff in the International Relations Center for the well and considerate arrangement of my research life in the foreign land.

I would thank all students who worked with me during the study, Shuang Wang, Yilin Zheng, Chunyuan Lan, Xiaoxuan Li, Zengyi Han, Yanyin Zhou, Takaaki Kubo, Yoshida Kentarou, and Naoki Higashi. I would also thank other members of CHEC for their great support.

I appreciate my parents, Weixun Li and Fengping Dong for their support and the deepest love. I thank my wife Mingfan Ma who gives me the heart-melt warmth and supports in my life.

LIST OF ACHIEVEMENTS

International Conferences

1. Li, Y., Sarcar, S., Zheng, Y., & Ren, X. (2021). Exploring Text Revision with Backspace and Caret in Virtual Reality. *Proceedings of the 2021 CHI conference on Human Factors in Computing Systems*. (Accepted, to be published)

2. Li, Y., Sarcar, S., Kim, S., & Ren, X. (2020). Swap: A Replacement-based Text Revision Technique for Mobile Devices. *Proceedings of the 2020 CHI conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3313831.3376217 (Orally presented at CHI 2020 Japan Chapter)

3. Jiang, X., Li, Y., Jokinen, J. P., Hirvola, V. B., Oulasvirta, A., & Ren, X. (2020). How We Type: Eye and Finger Movement Strategies in Mobile Typing. *Proceedings of the 2020 CHI conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3313831.3376711 (Orally presented at CHI 2020 Japan Chapter and CHI Nordic 2020)

Posters

1. Li, Y., Kim, K., Zheng, Y., Kubo, T., & Ren, X. (2020). Enhancing Text Input Efficiency in VR Applications. *International Workshop on Human-Engaged Computing (IWHEC 2020).* (Honorable Mention)

2. Li, Y., Ono, K., & Ren, X. (2020). Investigating Cognitive Strategies when Correcting Errors: From the Eye - Hand Movement Perspective. *International Workshop on Human-Engaged Computing (IWHEC 2020).*

3. Yoshida, K., Li, Y., & Ren, X. (2019). Eye-Hand Coordination in Error Correction Process during Typing on Smartphones. *International Workshop on Human-Engaged Computing (IWHEC 2019).*

4. Li, Y. (2017). Emergent Pen: Use Smartphones as Styluses. *International Workshop on Human-Engaged Computing (IWHEC 2017).*

Grant

International Exchange Grant (100,000 JPY) from Marubun Zaidan for ACM CHI Conference on Human Factors in Computing Systems (ACM CHI 2020).