2020

Master's thesis

The Strategies of Text Revision in Virtual Reality Environments

1235064 ZHENG Yilin

Advisor REN Xiangshi

February 28th, 2021

Information Systems Engineering Course

Graduate School of Engineering, Kochi University of Technology

Abstract

The Strategies of Text Revision in Virtual Reality Environments

ZHENG Yilin

Extensive research and related techniques and algorithms enable users to type with promising speed and accuracy in VR environments. However, there lacks essential consideration on the facilitation of text revision quality and performance, especially for the typing with virtual keyboards. In this study, we first did a broad and thorough literature review to point out the pain-points about the lacking of text revision in current VR text input research. We found that, for VR virtual keyboards, backspace is the dominant tool available for text revision without considering the text revision efficiency, cursor control is less deployed an developed in current VR virtual keyboards. Facing this, we further explore the possibilities of text revision designs based on different kinds of backspace and caret (cursor) control and implemented four VR text revision techniques to enhance text revision performance when typing with virtual keyboards and handheld controllers. We conducted a comparative user study to further evaluate users' text revision performance with our four text revision techniques. Results showed that using the word-level backspace and the continuous cursor control can enhance users' capabilities when revising text with virtual keyboards. This study can contribute the community of text entry with 1) a basic understanding on current research status of text revision in VR text input designs, 2) a series of feasible text revision solutions based on the backspace and cursor control, and 3) design guidelines and suggestions for

future designers for the development of novel text revision tools and techniques.

key words text input, text revision, virtual reality

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The popularity of Virtual Reality (VR) provides the potential of applying daily scenarios into VR environments. VR provides a solid hardware foundation for the development of efficient working environments such as virtual office work[30, 28] and remote classrooms [50]. In such cases, users need to deal with a heavy workload on text input (e.g., sending emails, composing reports, taking class notes). Therefore, when users interact in an immersive virtual environment, an effective text input method is needed to ensure text quality and satisfaction. Text input methods for VR should not only show fast typing speed but also their effectiveness handling various situations, such as typos (and grammar) correction and content rephrasing.

The advantage of the virtual keyboard different from the the physical keyboard is that it is more suitable [18] and portable [67] to deploy in VR applications with less environmental requirements (e.g., stable and flat surface to place the physical keyboard). Currently, to achieve better working quality in VR, numerous researches focused on making new techniques and designs for improving the typing speed and typo correction with virtual keyboards. However, these researches rarely mention that how users can revise the text in a VR system using a given tool or technology.

In the current VR text input research, text revision has not received enough attention, even if it is an important subtask to ensure content accuracy [45]. Imagine a scenario of writing an email in a VR system: after entering the text quickly (typo-free), the user still needs to revise the content (e.g., delete, insert or substitute) to ensure that

the content is accurate and meaningful. Currently, the backspace key is the main tool for text revision, and it is no exception to the virtual keyboard. However, we found that in the practial typing scenarios, the revision target can appear anywhere in the input content. In fact, in most related researches, the backspace key can only be activated at the end of the input content.

Looking at the situation above, there are three research questions: *(RQ1) Can current VR text typing methods handle comprehensive text revision requirements effectively? (RQ2) How to improve the efficiency of VR text revision based on existing tools and designs to improve current VR text methods? (RQ3) What are the strategies of text revision in VR environments?*

To address these questions, we first investigated the existing text revision tools used in the current VR typing technology and found three issues: 1) most researchers are more inclined to improve the VR typing speed with higher accuracy (no typo), but they rarely consider how to improve the efficiency of text revision;2) the current text input design based on the VR virtual keyboard does not include cursor(caret) control; 3) backspace is the only tool that can meet the basic needs of text revision without considering efficiency. Facing the founded problems, we combine current available revision tools to explore the VR text revision designs. Then, we further implemented four VR text revision techniques using virtual keyboards and handheld controllers. We evaluated their text revision performance through a comparative user study. Results show that the combination of word-level deletion and continuous cursor control achieves better text revision performance in VR. Finally, we further discuss the findings and summarize them into the strategies of text revision in virtual reality environments.

To conclude, this dissertation proposes the concept of our strategies of VR text revision may also explore novel tools and techniques beyond backspace and cursor, or to introduce solutions applied in other platforms (e.g., smartphones) and it provides

guidelines for the future VR typing method designers.

The writing of this thesis consists of six chapters, namely Chapter 1 Introduction, Chapter 2 Related Work, Chapter 3 VR Text Revision Design and Implementation, Chapter 4 Experiment, Chapter 5 Discussion and Chapter 6 Conclusion.

In Chapter 1 Introduction, a general description of the implementation of this thesis consists of background and overview. In Chapter 2, related research is presented that supports the theory of VR text revision. This chapter explains about current VR text revision methods and summarizing their attempts to facilitate text revision. In chapter 3 analyzes the vr text revision design and implementation of this thesis. This chapter explains the details of the design, and discuss the core technical challenges and related solutions for better implementing the proposed text revision techniques. In Chapter 4 the procedure of implementing comparative user study is presented to validate the design and evaluate VR performance with text revision techniques we proposed. In Chapter 5 the discussion based on the results of our experiment is presented. Finally, Chapter 6 provides our conclusion for our study.

Chapter 2

Related Work

This chapter contains descriptions of related work to this thesis problem. We report the development of VR technology and text input devices, then do a through analysis on current text input methods in VR, and we close this chapter by summarizing their attempts to facilitate text revision.

2.1 Development of VR Technology

2.1.1 A Brief History of VR

With the development of VR technology, all kinds of VR devices emerge endlessly. In the 1950s, American photographer Morton Heilig invented the first VR device: Senorama, was a huge, fixed device with 3D stereo, 3D display, vibrating seat and fan. It is considered as one of the earliest VR devices [34]. In 1960, Heilig submitted a design patent for the first VR glasses, which had a stereo display function. In 1968, American computer scientist Ivan Sutherland invented the prototype of the first head-mounted display that is the closest to the concept of modern VR devices. It realized the preliminary posture detection function through ultrasonic and mechanical axis technology, thus making the first VR system.

In the 1990s, VR ushered in the first craze, and major game companies rushed to launch their own VR devices. However, at that time, the display technology, 3D rendering technology, and motion detection technology were immature, so these VR

devices quickly disappeared from the market. Manufacturers were waiting for the further development of VR technology. In 2012, VR technology has made a breakthrough and Oculus Rift came out. In 2014, Google released its VR experience version solution: CardBoard, which enables people to experience the effects of a new generation of VR at a very low price. In 2015, HTC Vive was officially released at MWC. The following year, Sony released PSVR. Subsequently, major manufacturers began to develop their own VR devices.

With the development of portable PC and glasses display technology, more and more industries that are closely related to human life need to use VR devices, such as medicine, education, etc. Currently, the development of VR technology and devices is in a stable stage and human-computer interaction in VR is also a problem that cannot be ignored.

2.1.2 Interaction in VR

Virtual reality is generated by a computer and simulates a three-dimensional virtual space with input and output interfaces. It provides users with the simulation of vision, hearing and even touch to allow users to experience nature interaction in the virtual environment. Virtual reality technology provides a new way to view and manipulate three-dimensional data, injecting new technical elements into the human-computer interaction and equipment remote control system. Compared with the traditional human-computer interaction system in the past, the virtual reality system has three basic characteristics [81]:

1) Immersion: In the traditional interactive system, the results of the observation and processing from the outside of the computer system are observed, while the virtual reality system is immersed in the environment created by the computer system;

2) Interaction: In traditional interactive systems, the keyboard, mouse, and single-

dimensional digital information are used for interaction, while in virtual reality systems, it interacts with the environment of multi-dimensional information by interactive devices(such as multiple sensors).

3) Imagination: In traditional interaction, it is inspired by the results of quantitative calculations to deepen the understanding of things. In the virtual reality system, perception and rational understanding are obtained from the qualitative and quantitative environment to deepen concepts and germinate new ideas.

With the rapid development of virtual reality technology in recent years, it has been widely used in medical, transportation, education, entertainment, military, archeology and other industries. With the optimization of input and output devices at the technical level, virtual reality transforms the two-dimensional world into three-dimensional, and the interactive paradigm of two-dimensional graphical interface is not enough to satisfy more complex, more diverse, freer and broader interaction [81]. A new revolutionary way of interaction has emerged. Virtual reality interaction has always been the focus and hotspot of research, and there is still no unified consensus interaction paradigm. The human-computer interaction methods in the virtual environment can be divided into two categories, namely autonomously interactive actions and non-autonomously interactive actions. Autonomously interactive actions refer to interactions by people themselves, including hand interactions and somatosensory interactions. Non-autonomously interactive actions refer to interactive actions initiated by the machine and the environment. Users do not need to make special actions, such as eye tracking, EEG electromyography, head position tracking, facial expression recognition, etc.

The current human-computer interaction mainly relies on the WIMP interaction paradigm [81], and the interaction in the real world is mainly carried out through the five senses of humans (taste, sight, hearing, smell and touch). Virtual reality aims to completely simulate the interaction of the real world. Virtual reality aims to completely

simulate the interaction of the real world. However, due to the limitation of technological development, the current main interactive methods are still focused on autonomously interactive methods-through virtual reality devices to obtain interactive information in a metaphorical manner. The current mainstream virtual reality device includes two positioners, two handles, and a head display. The positioner is used to capture the position of the head display, the handle is used to capture the position of the hand, and the head display is used to transmit 3D scenes and capture head positioning and orientation.

2.1.3 Devices for VR

The common VR devices currently on the market include: hardware devices (such as HTC Vive), software devices (such as Unity, UE, and personal computers), eye trackers (currently, manufacturers that provide eye trackers for helmets include AppleLabs, Tobii, FOVE), headsets, etc. There are currently three mainstream forms of VR equipment, as follows.

PC + head display + handle + positioning device completes the calculation and rendering through the high-performance host, and then transmits the 3D image information to the head display through the transmission line, and finally complete the positioning of the head display and the handle of the positioning device. The advantage of this configuration is that the image rendering platform is stable and efficient, and the computing power is powerful; its disadvantage is that it is expensive and the mobility is limited by the length of the transmission line. Both HTC Vive and Oculus Rift support this configuration method, among which the Light-house technology used by HTC Vive belongs to laser positioning technology, and the positioning technology used by Oculus Rift is infrared active optical technology.

The mobile phone + helmet completes the calculation and image rendering

through the mobile phone and displays the image through the mobile phone screen, while the helmet is only responsible for fixing the mobile phone and enhancing the effect. The advantage of this configuration is that it is mobile and the price is relatively low. Its disadvantage is that the VR effect is poor and cannot provide precise positioning. Samsung Gear VR supports this configuration method.

The all-in-one PC directly integrates image rendering and calculation functions into the helmet, and also integrates sensing devices such as gyroscopes and optical lenses. The advantage of this configuration is that it has greater mobility and is reasonably priced; its disadvantage is that it lacks computing power, cannot support large VR systems, and has insufficient battery life. Storm Demon, etc. support this configuration method.

By analyzing the market research report, it can be found that the current mainstream VR devices in the market are HTC Vive and Oculus Rift [74]. Therefore, HTC and Oculus are dominant in the future development direction of VR positioning technology.

2.1.4 The Development Trend of VR

In the past 30 years, virtual reality technology has developed relatively mature and has a high degree of social acceptance. With the promotion of HTC, Facebook and other technology companies, virtual reality hardware platforms and software development platforms have begun to enter the market. At present, VR has become the next outlet after the Internet and smart phones. Technology manufacturers around the world are vying to enter the VR market, and VR-related research and applications are in full swing. At the same time, with the popularization of 5G and the rapid development of commercial GPUs, virtual reality technology is now being used in more and more fields. In the near future, virtual reality may reshape human life.

In the entertainment industry, VR movies will surpass IMAX and 3D movies, bring revolutionary changes to the movie industry, and make audiences feel immersive [70]. In navigation, the 3D geographic information system combines traditional GIS with virtual reality to form a realistic map scene to enhance the user's navigation experience. The current VR system lacks a unified standard in both software and hardware. Therefore, in the future, there will be a unified worldwide standard. The future VR hardware design will start from the human vision, hearing, smell and touch to collect the most basic human sensory information, and design a new interactive paradigm that can replace WIMP.

2.2 Input Devices for Text Input Tasks

With the increasing popularity of mobile electronic information management, the diversity of text input devices has been greatly developed. Text input is one of the most common and important tasks in human-computer interaction. Text input, especially short text input, is an unavoidable problem. For example, enter the user name and password in the game or software, enter and adjust the attribute parameters in the building or design model, enter the file name to open and save, and make short tags and comments. To solve the problem of text input, researchers draw on the simple solutions of traditional desktops and make special hand-held text input devices. With the advancement of algorithms, they have also developed text input through voice recognition, gesture recognition, and character recognition technologies. To better complete the text input tasks, it is necessary to select a set of appropriate input devices. There are many different types of input devices [2, 17, 20, 25] to choose, and some devices are more suitable for specific tasks than others.

2.2.1 Classification of Input Devices

With the development of human-computer interaction technology and mobile computing technology, input devices have greatly improved in form and interaction methods. For example, in the early mobile devices, keyboards were mostly used, and now it has developed into recognition (including handwriting, voice, image input, etc.), sensing (data gloves) and multi-channel human-computer interaction technology. As an input device, the design of the keyboard considers the size, efficiency, the number of keys and the size of the character. Due to the limitation of the number of keys and the size of the character, in order to achieve a certain character coverage, the keyboard needs to adopt different key combination modes. For example, the general mobile phone keyboard adopts a 3×4 keyboard layout, a total of 12 keys. Amal Sirisena [64] divided input methods into three categories: keyboard-based input methods, pen-based input methods, and advanced input devices.

In fact, input devices can be divided into different categories according to different forms, methods, purposes:

1) According to the key input efficiency and mapping relationship: the keyboard can be divided into one key and multiple characters, multiple keys and one character, and one key and one character.

2) According to the form of the keyboard, the keyboard can be divided into virtual keyboard and physical keyboard.

3) According to input mode and operation purpose, it can be divided into pointing input device and text input device.

2.2.2 Common Text Input Devices

Based on the above classification, input devices can be divided into pointing input devices, text input devices, and other input devices such as image capture. Here we mainly introduce and analyze some common text input devices [63, 66].

The **physical Keyboard** is the most commonly used text input device for computers. It is a typewriter-style device which each button typically represents one character and there are many different keyboard layouts available. From the earliest typewriter to QWERTY layout [56], designed by Christopher Sholes, which was patented in 1869 [46], is a standard English keyboard layout for the physical keyboard (Fig. 2.1).

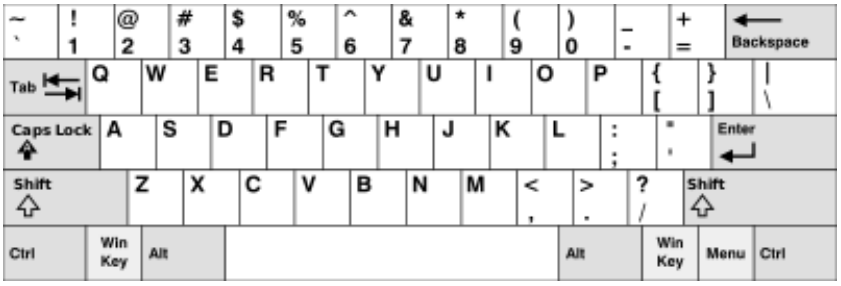

Fig. 2.1 An ISO standard layout for QWERTY Keyboard [14].

After that, although researchers also launched some competing products, such as Dvorak Simplified Keyboard (Fig. 2.2)and Split Keyboard (Fig. 2.3). Stan J and Stephen E examined the records of typing experiments in the ergonomics study, as well as the records of comparison experiments between different keyboard designs in history, and found that other layouts did not significantly improve typing efficiency compared to the QWERTY standard keyboard [47]. Until now, the QWERTY keyboard is still the most common and popular keyboard that people use on computers, mobile phones and other smart devices.

A **touchscreen** is a both input and output device and normally layered on an electronic visual display of an information processing system. A user can give input or

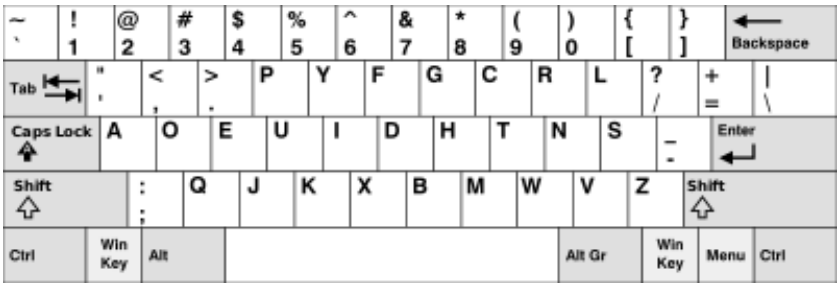

Fig. 2.2 Layout of Dvorak Simplified Keyboard [13].

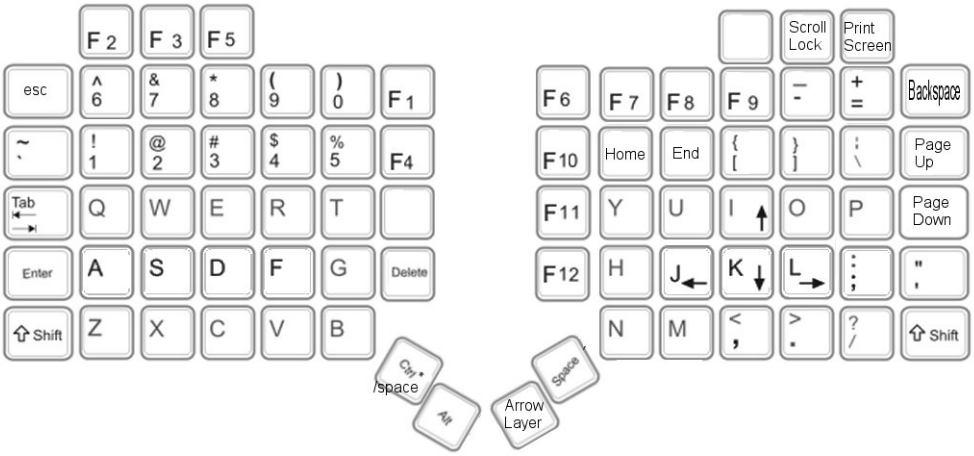

Fig. 2.3 A common used split Keyboard [79].

control the information processing system through simple or multi-touch gestures by touching the screen with a special stylus or one or more fingers [69].

The **Soft keyboard** is not a physical keyboard. It uses a touch screen technology to simulate the shape of the keyboard on the screen and realize the input, which can be operated like a normal keyboard (see Fig. 2.4). The size, shape and position of the soft keyboard can be adjusted arbitrarily. A notable feature of the soft keyboard is that it can be changed with the user's personal preference. The soft keyboard has two problems that have not been solved for a long time. First, due to the lack of tactile feedback, users must always look at the display to confirm that they are typing correctly. Another problem is that the soft keyboard occupies the limited display resources of the display

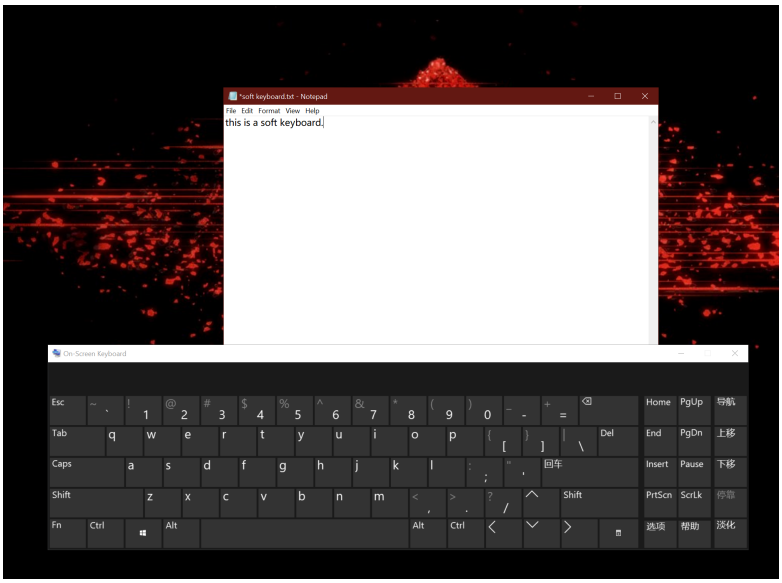

Fig. 2.4 An example of a soft keyboard.

screen, which affects the computer's image output and user perception.

ITU-T text input (phone keyboard): With the popularity of text messaging , text input with mobile phones has gained use. Each key on the mobile phone keyboard can contain multiple characters and can be accessed through multiple key presses. It is usually used in conjunction with predictive text input. Although once very popular, with the widespread use of touch screens on smart phones, this input method has gradually been replaced.

Handwriting Recognition allows users to use a touch screen device, much like a notepad on which they can write on the screen without a keyboard. Handwriting recognition mainly uses either optical character recognition, which uses an optical scanner to scan the words written by the user to determine the best suitable match, or uses a pen-based computer interface to track the movement of the user's pen tip to write.

As far as proficiency is concerned, handwriting recognition has significant charm in text input. Ideally, a handwriting recognition system is as simple as writing on paper. Anyone can use this device without additional learning, and the input speed is equivalent

to a QWERTY beginner. There are two input methods in the handwriting recognition system. First is to write directly on the screen. Although this input method does not take up a lot of space, the hand will cover part of the screen when writing, which is not conducive to continuous interaction. The second way is to write on a special tablet. It avoids the problem of hand obstructing the display when writing.

From the perspective of recognition rate, handwriting recognition is still not perfect. It is generally agreed that handwriting recognition can be widely accepted only when the accuracy of handwriting recognition reaches or exceeds 97%. The advantage of the handwriting recognition system is that it has the function of real-time information collection and processing. For example, because the system can monitor the writing trajectory of the handwriting pen when writing, the system can distinguish between very similar character such as v's and w, which will help the handwriting recognition system to achieve the desired accuracy. But for some special characters, such as artistic characters, the handwriting recognition rate is only 87% to 93%, which is far lower than the acceptable accuracy rate of 97%.

Voice recognition is a system that allows users to use their own voice to send messages, make calls and so on. It requires the speaker to speak slowly, clearly and to separate each word with a short pause, then it can work by analyzing sound and converting it into text. This text input method can replace or supplement other input devices to provide a fast text input method without a keyboard, and can help people with various disabilities. Some of the most well-known systems are Apple Inc.'s Siri and Cortana which are developed by Microsoft.

From some aspects, voice recognition input is an ideal text input method, it can be used in almost any environment and occasion. However, voice recognition input often limits the environment and requires a relatively quiet environment. In addition, for meaningless characters, users need to pronounce them word by word. Another

problem with voice recognition systems is privacy. Privacy is a big problem in the work environment. Users do not want to make a sound in certain situations, such as entering a password. Therefore, voice interaction requires more research to adapt to real-world use situations. At the same time, in the work environment, voice control is very useful. Users can free their hands to do other things when working, such as accessing information databases to help complete some tasks and other text input tasks. Voiced text input is ideal in these situations.

At present, there are still many technical obstacles in the development of voice recognition, such as low recognition accuracy and low input efficiency. Voice recognition is mostly used in the following situations: 1) It can be used to support devices that are too small and not suitable for other types of interfaces; 2) It can play a good role in places where the keyboard is inconvenient to use; 3) In some public facilities(such as ATMs), automated voice recognition can replace manual input; 4) It can make it easy for untrained users to use the computer; 5) It is very beneficial to users with visual impairment or other physical impairments.

Virtual Keyboard is similar to a mechanical keyboard that does not use physical keys. It can be implemented in a virtual reality environment, where a virtual keyboard that simulates a real keyboard is placed. To complete the input task, users need to touch the characters on the virtual keyboard that they see. The virtual keyboard can contain all standard keys including letters, numbers, symbols and system keys while does not require the use of a physical keyboard. These keys can either be selected by a mouse or other pointing devices, or can also be combined with VR devices. The virtual keyboard most often uses the standard QWERTY layout, and it provide an alternative mechanism for disabled users who cannot use a physical keyboard.

As a main text input method, using a virtual keyboard with QWERTY standard layout to complete a text input task in a virtual reality environment consists of two

steps. The first step is indexing, that is, to locate the key of the target character in a certain way; the second step is to confirm, that is, to confirm that the key is entered by a certain instruction, so it is also called hunt and peck [80].

At present, many virtual devices can assist users to use virtual keyboards. Oculus Rift, HTCVive, and PS VR all tend to use their own controllers and headsets, because users need to use them all the time in a virtual environment. Therefore, one of the most common ways is to use a headset with a display to observe the virtual keyboard in the virtual environment, use the controller to select letters one by one, and then press the buttons on the virtual device to enter the virtual keys.

On this basis, many researchers and developers have made improvements. Google Daydream Lab helped HTC Vive build a drumstick controller, a more rhythmic drumstick keyboard for input, which makes the input more natural like playing drums. Compared with key clicks, the drumstick keyboard percussion action is a more direct operation [42].

The text input method based on the QWERTY virtual keyboard is still the mainstream in the field of virtual text input today. Researchers have proposed a variety of improvements: 1) Indexing by handles, rays, touchpads, or even bare hands, gaze, etc.; 2) Confirmation by keystrokes, hand movements such as pressing or gestures, etc.; 3) Proposing predictive algorithms and increased tactile feedback to improve user interaction experience and performance.

Nowadays, most VR devices lack an intuitive input mode, which cannot be combined with a high input rate to provide users with complete control over their virtual workspace. Therefore, in a virtual environment, it is important to design a good text input experience for users.

2.3 Physical Devices for Text Revision

Due to the popularity of VR devices, correspondingly, the requirements for text input have also increased in the VR environments. Physical keyboards (with QWERTY layout) are powerful devices used for VR text input. On the one hand, head-mounted displays (HMD) are critical to creating an immersive experience, so users can no longer see their hands or traditional input devices, such as keyboards and mice. On the other hand, ordinary VR input devices such as motion-tracked handheld controllers cannot provide the same level of haptic and tactile feedback as a keyboard. However, due to the occlusion of HMDs, typing with physical keyboards in VR cannot achieve equivalent performance as that in the real world.

To enhance the VR typing performance with physical devices, researchers attempted to explore their understanding about the influence of factors, such as hand representation [26, 40], keyboard representation [52], and blending of reality in VR [48] etc., on users' typing performance. Similar to the text revision behavior in our real-life scenarios with physical keyboard, we can also use the backspace and arrow keys on it to correct typos and revise unintended words. Although there exist previous designs that include arrow keys in their designs [5, 27, 35, 39, 54, 58, 59], no further description or discussion was founded on the use of arrow keys dealing with the scenarios such as typing and revision. Instead of using arrow keys, Walker et al. [70] used auto-correction features as the replacement of the backspace (they disabled the backspace key and prevent participants from using it).

Except for physical keyboards, researchers also explore the feasibility of using other devices such as data gloves, custom hardware, or IoT devices for text entry in VR environments. By leveraging the rotation and the pinch gestures, Pinch Keyboard [6, 7, 23] enable users to enter text when wearing the data gloves on their hands. To perform

the function of backspace, users can make a pinch gesture with two ring fingers to realize the function. Cursor control, whereas, is not included in its design. KITTY [43, 53] uses all finger joints to reflect different keys. Users can use their thumb to perform the pinch operation on different joints on the hand and map it as the character input into the system. Cursor control mechanism for KITTY is missing. Although the delete key was included in their prototype, there lacks an essential description on how to use it to correct typos. Wu et al. [73] used data gloves to provide and enhance the haptic feedback when pressing the keys on the virtual keyboard. To finish the revision, Wu et al. allowed users to revise text with the help of the backspace key and arrow keys. We also found a prototype, named K3 [10], which serves more as an interaction integration platform. Users can put different buttons on it to realize various forms of interactions. In their paper, they also mentioned the potential of using K3 for text entry. However, no further information or detailed description was founded for the text entry design.

2.4 Virtual Devices for Text Revision

Virtual keyboards are intangible ones that sometimes flow in the mid-air of the virtual space, users can type on it with their hands or through controllers. Virtual keyboards are regarded as suitable [18] and portable [67] to deploy in VR environments because that users do not need to find a solid platform to put their physical devices. Backspace is commonly included in the previous design to support the potential text revision requirements. It is surprising that, through our summary of previous VR text input research with virtual keyboards (see Table 2.1), no current text input research with virtual keyboards considered cursor control mechanism, solutions, or techniques in their designs. Even worse, most of researches did not mention or discuss about the cursor control and its importance. In other words, backspace in existing research [3,

4, 11, 12, 16, 36, 29, 21, 37, 38, 49, 55, 57, 60, 61, 65, 67, 72, 76, 78] can be regarded as the main and the dominant tool available on the virtual keyboard for text revision. When implementing the backspace, the function was inherited from the conventional backspace on the physical keyboard. In detail, when pressing the backspace, the system will both delete one character and move the cursor back to the distance of one character [71]. Previous research [41] pointed out that backspace is available and effective for quick corrections. For instance, delete the characters that are near the current cursor position [45]. Without caret (cursor) control, it is difficult and time-consuming to deal with the scenario where the target to revise or to delete is far away from the cursor. Users need to repetitively use the character-level backspace and re-enter the characters (as the victims of having no cursor control) that deleted unnecessarily during the revision process.

There exists a previous research on enhancing the backspace functionality on smartphones without cursor control. Facing the error correction process during typing on smartphones, Arif et al. [1] combine the Levenshtein distance [44] to locate the error position and the swiping gesture to cut and store the right characters after the typo position. After the regular correction, another swiping gesture can be used to recover those cut characters back. However, this solution cannot be effective when facing conditions where the Levenshtein distance algorithm did not find the error, especially for text revision, the task that usually contains no typos but unintended meaning according to users' flexible intention.

Function of the backspace can also be upgraded from character-level to word-level to achieve effective deletion performance. RotoSwype [31] and GestureType [77] allow users to delete the word just entered. RingText [75], DwellType [77], and TapType [77] enable flexible deletion (at the character-level when entering and at the word-level when editing) according to the content just entered. Although the aforementioned

research improves the power of backspace, its usefulness is still be limited without the help of cursor control tool, especially when dealing with more sophisticated text editing conditions.

2.5 Text Revision as More General “Correction”

Techniques such as en/decoders [44, 77], auto-prediction, and auto-completion algorithms has been extensively proposed to help users to type text quickly without grammar issues. Those work can ensure the text input accuracy in the spelling and grammar level, which enables us to discuss the questions that are more complicated and more commonly encountered. In our daily life with text, users typo-free and grammar correct is only the basic requirement for all typist. Beyond that, it is more important to make sure the entered text can accurately and properly express the intention of the typist [45]. After quickly entering the text, more time will be used to finish the proofreading such as deleting redundant words, replace improper deceptions, etc. For such conditions, till now, we do not have powerful tools except backspace and arrow keys (there even have no arrow keys for VR text revision with virtual keyboards).

2.6 Summary

Text input is a task that helps users to communicate among others with the form of text. Promising typing speed and accuracy are vital but not the decisive factors [18] to determine whether one text input technique is useful. We summarized 41 research articles (14 for physical devices, 27 for virtual keyboards) from the previous two decades about text input research. Table 2.1 shows that there is still a long way to develop for current text research in VR environments to achieve the equivalent usability as that in other platforms. Especially when facing the text revision problem, current research

Table 2.1 Summary of current text input research from the perspective of backspace, cursor control, and text revision. We summarize 14 research for physical devices and 27 for virtual keyboards in the last two decades.

Perspective	Input Device	Included	Not Included	Not Mentioned
Backspace	Physical Devices	12	1	1
	Virtual Keyboards	25	2	0
Cursor Control	Physical Devices	6	6	2
	Virtual Keyboards	0	27	0
Text Revision	Physical Devices	3	0	11
	Virtual Keyboards	4	0	23

on virtual keyboards did not include cursor control, and text revision consideration is seldom mentioned or discussed. This gap leaves us the opportunities to explore the potential solutions and techniques to enhance users' text revision performance in VR with virtual keyboards.

Chapter 3

VR Text Revision Design and Implementation

In this chapter, we mainly discuss the following three points: 1) we illustrate the design space we proposed for the systematic understanding and exploring the possibilities of text revision solutions in VR, 2) based on the design space, we propose four VR text revision technique candidates that combine different options of using backspace and caret (cursor) based on handheld controllers, and 3) we will discuss the core technical challenges and related solutions for better implementing the proposed text revision techniques.

3.1 Combining Current Available Revision Tools

In current VR systems, the most mature text revision tool is arrow keys and backspace on the physical keyboard (this solution happens when users type on a physical keyboard in VR). However, for the conditions without available physical keyboards (especially for virtual keyboards), we found that there is a lacking of those essential text revision tools after we summarized through the current literatures. This situation make us consider: whether we can use the caret (cursor) control mechanism (e.g., arrow key is one of the representatives of cursor control mechanism) and backspace to enhance text revision performance for the typing activities with virtual keyboards.

In detail of VR text revision with virtual keyboards, we found that commonly applied text-revision related tool is the backspace (some of the research even did not include backspace in their design). Caret (cursor) control mechanism is missing in the existing designs. Therefore, based on the current available tools, the first step is to re-introduce the caret (cursor) control mechanism to the text revision design process for virtual keyboards. If there is no caret (cursor) control mechanism involved, the cursor would lose its flexibility of moving among characters and thus be fixed at the end of the string. Users may have to use the backspace only but repetitively to finish the revision.

We also notice that there are different options and functionalities that already applied on other platforms regarding the backspace and caret (cursor) control. Therefore, through our further investigation on the possibility of combining different kinds of backspace and caret (cursor) control mechanisms, we choose the two feasible options for backspace and caret (cursor) control mechanism. For better understanding, we visualize our choices and their combinations in the form of design space as shown in Fig. 3.1. We put two axis into our consideration: backspace granularity and caret (cursor) control continuity. We will explain in detail in the following subsections.

3.1.1 Character-level Backspace vs. Word-level Backspace

Backspace usually represents the function of removing character(s) from the existing text content. Through literatures and current commercial products, we choose character-level and word-level as two options to consider the efficiency of the backspace for text revision. It should be noted that text revision is seldom mentioned in the previous research and backspace is not directly mentioned for text revision (it is more regarded as error/typo correction). However, in daily text-related interactions, backspace is used to deal with both error correction and text revision. For character-level backspace, it is the most common and default form of backspace that

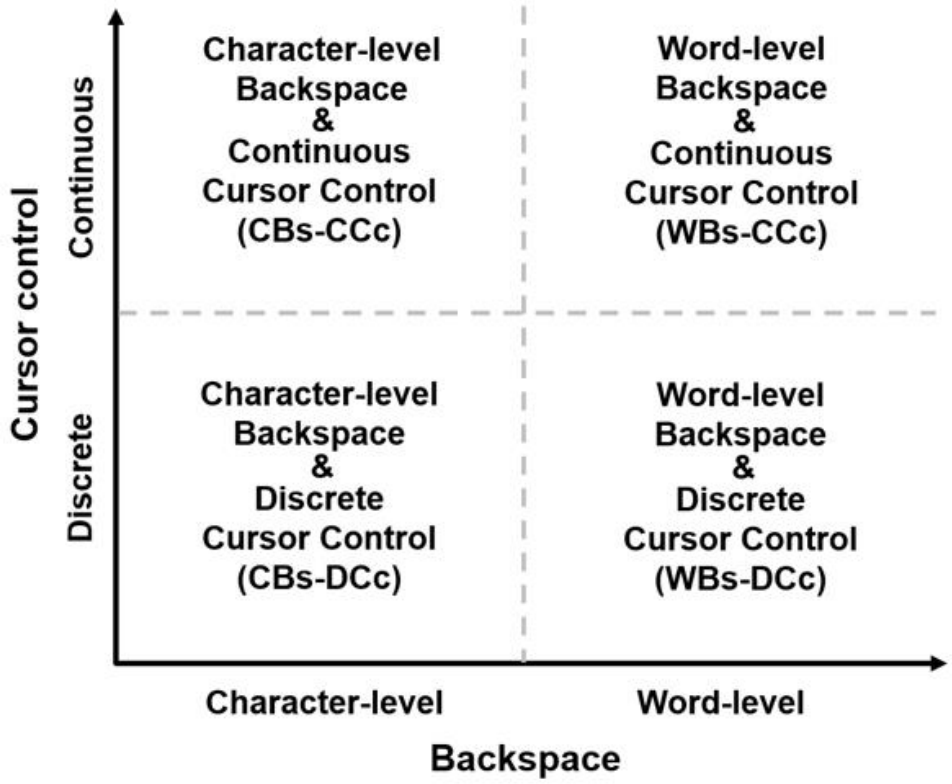

Fig. 3.1 All combinations and their abbreviations for different backspace and caret (cursor) control functions for VR text revision with virtual keyboards.

already applied in almost all text input commercial products. When the character-level backspace is triggered, one character/symbol before the caret (cursor) will be deleted along with the backward of the caret (cursor). For word-level backspace, the granularity of deletion changes from one character to multiple characters. Word-level backspace is also a common function in current typing systems. For instance, in the Windows operating system, users can perform the shortcut of Ctrl + backspace to realize multi-character deletion.

3.1.2 Discrete vs. Continuous Cursor Control

Similar to backspace, we also have multiple choices when considering the function of cursor control. Based on the consideration of various approaches to navigate the cur-

sor to select keys in selection-based VR text input techniques [67] and users' movements when performing actions [8], we found that the most common forms of caret (cursor) control contain two main types: discrete and continuous. The difference between discrete and continuous cursor control is about the continuity of one single execution of the manipulation. For instance, when facing the situation of moving the caret (cursor) to the 10-character left from the current caret (cursor) position, users need to press 10 times of the left arrow key on the virtual keyboard for discrete caret (cursor) control, or users can use their finger to directly control the caret (cursor) to the position for continuous caret (cursor) control (with only one time of manipulation).

3.2 Four VR Text Revision Techniques

Based on the two kinds of backspace functions and two types of caret (cursor) control mechanisms, we can combine them and therefore produce four combinations (as shown in Fig. 3.1) as the potential candidate solutions to enhance users' text revision performance. For better validating their feasibility and understanding users' performance differences using different solutions. We decide to implement the four solutions. During the realization process, we choose to use the handheld controllers and VR headsets to finish the implementation of four VR text revision techniques.

3.2.1 Handheld Controller as a Powerful Tool

Current commercial VR products equip portable handheld controllers as users' default input devices. The lanyard on the controller makes it possible and flexible for users to switch between free-hand and controller-based interaction by simply dropping and grabbing controllers. Meanwhile, handheld controllers also provide various interaction patterns through physical buttons, triggers, gyroscope sensors (for detecting controller

rotation) and touchpads. For VR text input, handheld controllers (and joysticks) are mainly used to locate and select keys. Additionally, researchers also leverage the controller swiping in the mid-air or the finger movement on touchpads to enter text with word-level gestures.

Bare hand movement can be detected by motion tracking systems and leveraged to implement free-hand typing with virtual keyboards. Although bare hand typing frees hands from holding extra devices, users still need to face challenges during typing: 1) Users' forearms need to hover in the mid-air without sufficient support, which makes users feel fatigue (especially for long-term use) and 2) To extend the functionality, users perform gestures to trigger commands (e.g., copy, paste). Users need time and effort to learn and remember multiple gesture-command mappings before mastering them. Head-based and gaze-based typing allow users to type on keyboards with few limb movement (which is especially beneficial for the impaired). Users need to drive their heads and eyes to search among keys with higher requirement of precision. From the perspective of ergonomics and neuromechanics, however, frequent and restricted head and eye movement bring fatigue and potential discomfort (e.g., dizziness and dry eye syndrome) that influence the interaction experience in VR.

3.2.2 Implementation Details

Generally, it is not possible to have independent text revision techniques because 1) text revision depends on text entry techniques and tools, and 2) text revision also include part of the process of entering text, such as replacing words or putting some missing words into the sentence. Therefore, we also need to consider the use of VR text entry tools that help us to finish the text revision study. We choose to implement our proposed techniques based on a virtual keyboard with the standard English Qwerty layout. For the flexibility of navigating the caret (cursor), we visualized the caret (cursor) and

enable the function of allowing users to control the caret (cursor) on the textbox widget (as most of previous research did not include caret (cursor) into their consideration). Additionally, to both simplify the implementation process and to decrease the influence from extra factors, we use the physical buttons, triggers, and the touchpads on handheld controllers to develop all four techniques.

Character-level backspace and discrete cursor control(CBs-DCc) is the most fundamental combination inherited from the backspace key and arrow keys on physical keyboards. In the technique of CBs-DCc, users control the cursor by pressing left or right part of the touchpad. Users can also use the physical button on the topside of the handheld controller to delete one character per pressing.

Character-level backspace and continuous cursor control(CBs-CCc) learns from the physical keyboard and the laptop touchpad. With CBs-CCc, users can perform the sliding with their finger on the controller touchpad to move the caret (cursor) left or right continuously before lifting up their finger. Similarly, users can also use the physical button on the topside of the handheld controller to delete one character per pressing.

Word-level backspace and discrete cursor control(WBs-DCc) allows users to press the button on the handheld controller to delete multiple characters at one pressing. Users control the cursor by pressing left or right part of the touchpad.

Word-level backspace and continuous cursor control(WBs-CCc) allows users to navigate the cursor to the position of the revision target via sliding with their finger on the touchpad. Users can execute the word-level (with multiple characters) deletion by pressing the button on the topside of the handheld controller.

3.3 Technical Challenges and Solutions

We choose Unity 3D environment with C sharp programming language to implement the four techniques and the experiment system in VR with the help of SteamVR, a mature development kit for VR application development with HTC Vive. During the development process, there were five main challenges need to solve to realize the intended techniques and the system: 1) how to detect key pressing events from the controller manipulation, 2) how to realize the button pressing behavior with virtual keys and the virtual mallets, 3) how to record users' behaviors during revisions, 4) how to implement the word-level deletion, and 5) how to realize the continuous caret (cursor) control with the sliding on the touchpad.

3.3.1 Event Detection

To detect the events when using the controller, such as pressing the buttons, and sliding your finger on the touchpad, we need to bind the script (with the specific event detection and treatment codes in it) to both controllers, as that shown in Fig. 3.2. With the binding, all manipulation behaviors can be sent to the script for further treatments. Then, from the code level, we need to get the tracking object (the controller detected by the VR system) by using the function of GetComponent <SteamVR_TrackedObject>(). Then we can assign the VR tracked object (e.g., named device) to an instance of SteamVR_Controller.Device object through device = SteamVR_Controller.Input((int)trackedObject.index). The purpose of this step is to make sure that we can get well-prepared handheld controller object for us to handle different controller-related events. After that, we can put various functions in the default function of Update() to realize the real-time event detection. For examples, 1) using device.GetPressDown(SteamVR_Controller.ButtonMask.ApplicationMenu) to

handle the pressing of the button on the upper side of the touchpad, 2) using device.GetPressDown(SteamVR_Controller.ButtonMask.Touchpad) to detect the pressing of the touchpad, and 3) using device.GetTouch(SteamVR_Controller.ButtonMask.Touchpad) to detect the touch behavior on the touchpad.

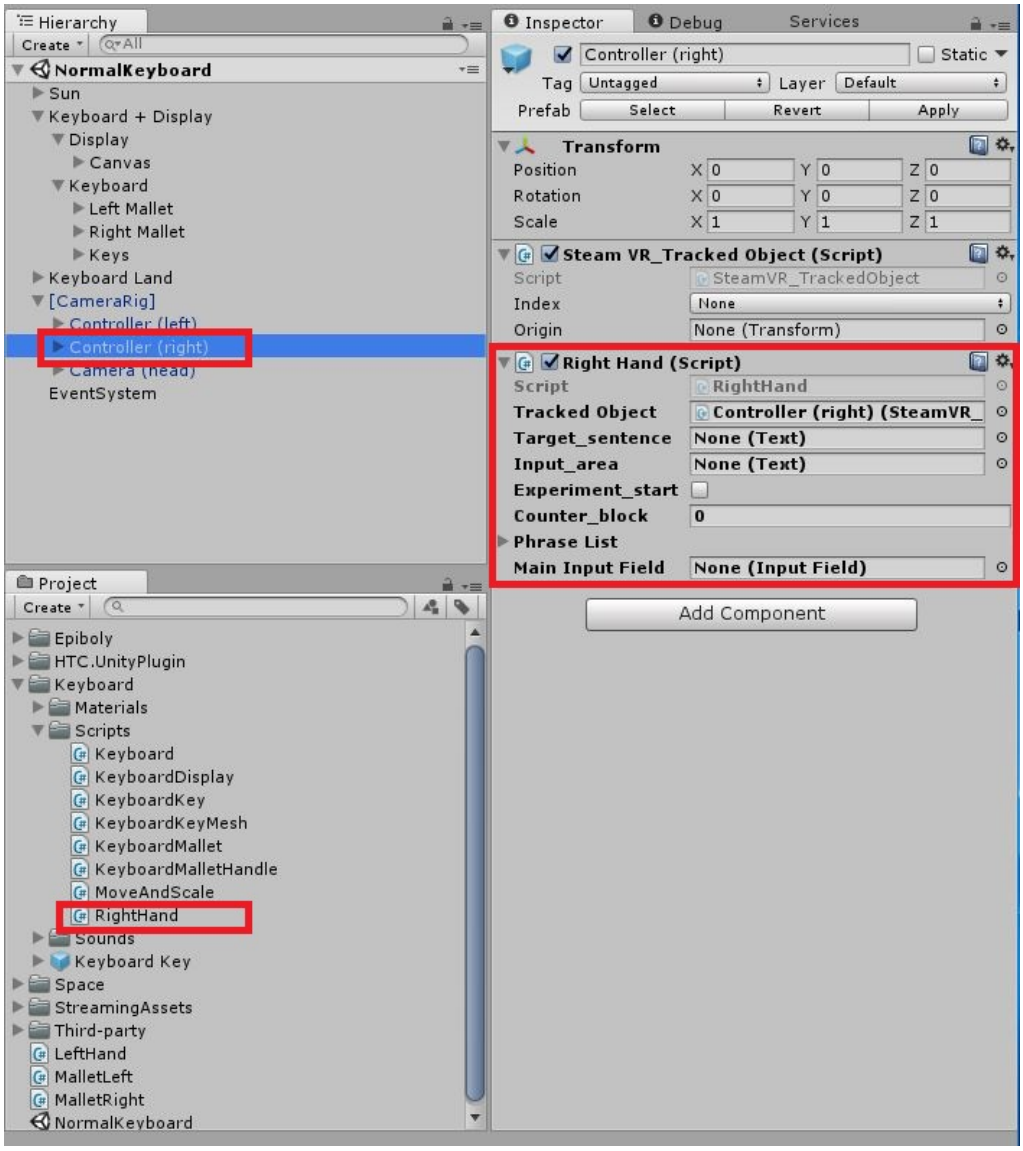

Fig. 3.2 The script named RightHand is binded to the handheld controller (as shown in red frames).

3.3.2 Button Pressing with Virtual Mallets

In the development, we chose to use a virtual mallet (extended from the top of the handheld controller) and use it to strike the virtual keys for selection, just like hitting the drum surface with a real mallet. To realize this, we consider the head of the virtual mallet and the virtual key as a sphere and a cuboid. Then we give them with the collider feature and make them available for detecting the collision of different objects. Then, we code the script binded to the virtual mallet under the function of OnTriggerEnter (COllider other), which we can get the specific object that makes collision with the virtual mallet. The real-time position of the virtual mallet was detected and recorded within the function of Update(), where functions in this Update() will be ran every frame. After getting the object made collision with the mallet, we can further get the label on the object (e.g., the letter shown on the button) and send it to the textbox to show the selected key.

3.3.3 Data Recording

For better analysis of the text revision behavior, we need to make sure that all available data and behaviors can be recorded timely and accurately into the computer system. We chose to use the text log to record every single interaction that can be detected by the experiment system. After getting the events such as physical button pressing, character input, and touchpad sliding, those events will be transformed into one line of text containing specific information (e.g., timestamp, event name, character, cursor position, text in the textbox, etc.) according to the event types. Each line of the text will be transformed into a byte array and wrote into a txt file in the computer using the Filestream class and its Write (byte array, offset, count) function.

3.3.4 Word-level Deletion

As we include cursor into our design, we need to consider the cursor position when implementing the word-level deletion. In detail, the system will count the cursor position to confirm which word is before the current location of the cursor, then perform the multiple-character deletion function. In order to delete multiple characters, we used the virtual keyboard API (in user32.dll with the entry point of keybd_event) to simulate the trigger of character level of backspace multiple times. In detail, Keybd_event(8,0,0,0) represents the behavior of pressing down the backspace key, while Keybd_event(8,0,2,0) represents the release of the backspace pressing.

3.3.5 Continuous Cursor Control with Sliding

If we want to use the finger sliding to control the caret (cursor), we need to do the real time sliding movement detection on the touchpad with device.GetTouch(SteamVR_Controller.ButtonMask,Touchpad) in the default function of Update(). Using the GetTouch function is to get the real-time finger position on the touchpad. However, the frequency of this function is very high, which makes it too sensitive for users to control the cursor. Therefore, we set a time interval to periodically get the finger position with a relatively longer period instead of every frame. Then we can compare the position offset between the current position and the previous position of the finger on the touchpad. With this information, we can further map this direction change to the position adjustment of the caret (cursor).

Chapter 4

Experiment

After the implementation of the experiment system, we need to investigate whether our proposed techniques can be capable of handling VR text revision and enhance users' text revision performance in VR. Therefore, this section we illustrate details of the comparative study we conducted for evaluating the four proposed text revision techniques in VR environments with handheld controllers.

4.1 Participants

We got 16 volunteers to take part in our comparative user study. Nine of them are female and seven of them are male. The average age of them was 24.81 years old with the standard deviation of 2.4. All participants had no physical health problems. All of them are with normal vision (or corrected-to-normal by wearing glasses). 15 of the participants report themselves as right-handed and one as left-handed. All participants had no previous experience to finish any text-related tasks or interactions in VR environments. They are all familiar with the keyboard with the QWERTY layout. As for their English capability, they reported themselves capable to read and write fluently in English.

4.2 Apparatus

We used HTC Vive (including the head-mounted display and handheld controllers) to develop our experiment system. The experiment system was implemented and ran on a desktop PC (running the Windows 10 operating system) with an intel i7-3770 CPU, 16GB RAM, and NVIDIA Quadro K4000 graphic card. We used two HTC Vive optical trackers to set a $2.5m(length) \times 2.5m(width) \times 2m(height)$ spatial motion capture area. In this area, all users' motion can be detected, captured, and recorded by the experiment system. We chose Drum-like keyboard [24] as the text input technique for our experiment with the following reasons: 1) text revision techniques cannot exist individually without a proper text input method and 2) compared with other VR text input techniques, Drum-like keyboard performs better in the perspective of typing speed, typing accuracy, as reported in the literature [3]. Drum-like keyboard applied the metaphor of beating a drum in the scenario of VR text input. In detail, the intangible keyboard floats in the spatial area, users can select the character by hitting the intended key with a virtual mallet that extends from the top of the handheld controller. We chose Cutie keys [15] to implement Drum-like keyboard because Cutie keys is open-source and easy to deploy in VR environments. The experiment system was developed in the environment of Unity with the version number of 5.6. For the word deletion function, we got ideas from the existing attempts [62] that already applied in current computer operating systems to delete multiple characters at a time.

4.3 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 [68] 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 [19, 67], and (3) compared with the phrase set from MacKenzie and Soukoreff [51], 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 put only one revision target for each sentence. We learned from the categorization of character-level error types [22] 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 [45]. 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 [82]. Please see the example sentences in the Appendix A. On average, each target contains 5.08 characters (SD = 1.79).

4.4 Design and Procedure

We use a within-subjects design in our experiment to investigate the four proposed VR text revision techniques named CBs-DCc, CBs-CCc, WBs-DCc, and WBs-CCc separately. Under the protocol of a within-subjects experiment, every participant is requested to use each of the four proposed techniques to finish the text revision task. For each techniques, the participants need to finish the text revision task with 30 sentences. It means that every participant needs to revise 120 sentences during his/her experiment period. The order of using four text revision techniques has been counterbalanced among participants with a Latin Square. Based on the description above, the total number of text revision behaviors was: 16 participants $\times$ 4 text revision techniques $\times$ 30 sentences

per technique = 1920. We randomly chose every 30 sentences from the corpus for each technique. We also made sure that the previously chosen sentences would not appear in other techniques. In every 30 sentences, we put a certain number of various types of revision targets (e.g., character-level and word-level insertion, omission, and substitution targets) and randomized the order of their appearance during the revision process.

We first introduced the purpose of the experiment 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 [33] and System Usability Scale (SUS) [9] to evaluate workload and subjective preferences towards all four techniques.

At the initial stage of the experiment, we described the aim and rough procedure of the following experiment to our participants and then demonstrated the four proposed

VR text revision techniques and different types of revision targets to them. After that, we assisted participants to wear the head-mounted display and guided them to sit in the chair (in the motion detective zone). We then helped the participants to adjust the height and orientation of the virtual keyboard shown in the head-mounted display. After the adjustment, all participants would have a 15-minute period of practice to further learn and get used to the Drum-like keyboard and the four proposed text revision techniques. Participants were allowed to practise all revision techniques on demo sentences before the formal experiment. We recommended all participants to use the physical button and the touchpad of the controller on his/her dominant hand.

For each trial of text revision, two sentences will appear on the head-mounted display, one is the standard sentence and the other is the sentence with one revision target in it. Participants are requested to revise the target as fast and accurate as possible with the appointed technique. After the revision, the revised sentence can be submitted by pressing the grip button on the controller. We use the strict protocol to make sure that the new trial would not appear unless the participant successfully revised the current sentence (two sentences should be matched before submission).

For each participant, the formal experiment would last about 1 hour. After revising 30 sentences, there would be a 5-minute break before next 30-sentences revision. After revising 120 sentences, all participants are requested to fill the NASA-TLX [33] and System Usability Scale (SUS) [9] to evaluate workload and subjective preferences towards all four techniques. All comments from participants during or after the experiment would be noted for the following analysis.

4.5 Results

Before the formal analysis, we removed 4.79% (92 items of data, including outliers and trials that mis-interrupted by the mis-trigger of the VR system menu) percent of data from the collected dataset. We then check the normality of the remaining data and found that the data set did not obey the normal distribution. Therefore, 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 [32] 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. We mainly investigated and reported the influence of four techniques on participants' VR text revision performance in the perspectives of correction time, operation per character, number of backspace, backspace time, number of caret (cursor) control, caret (cursor) control time, and subjective preference (through SUS score and NASA-TLX).

Operation per character counts the average number of executions (e.g., character selection, different kinds of backspace, and caret (cursor) control operations) that required to revise one character during the sentence revision process. Fig. 4.1 shows the statistical results of operation per character for four proposed text revision techniques. Results showed that CBs-DCc needs the largest number of operation to revise one characters (with the mean of 4.81), while WBs-CCc uses the least (with the mean of 1.52). We found a significant effect of technique on the operation per character with a Friedman test ($\chi^2(3) = 682.68$, $p < .001$). We also ran a post-hoc test and found significant differences among all technique pairs (all $p < .001$). Fig. 4.2 shows the operation per character when participants revised targets with different distances. Results showed that techniques with continuous caret (cursor) control can decrease the operation per character.

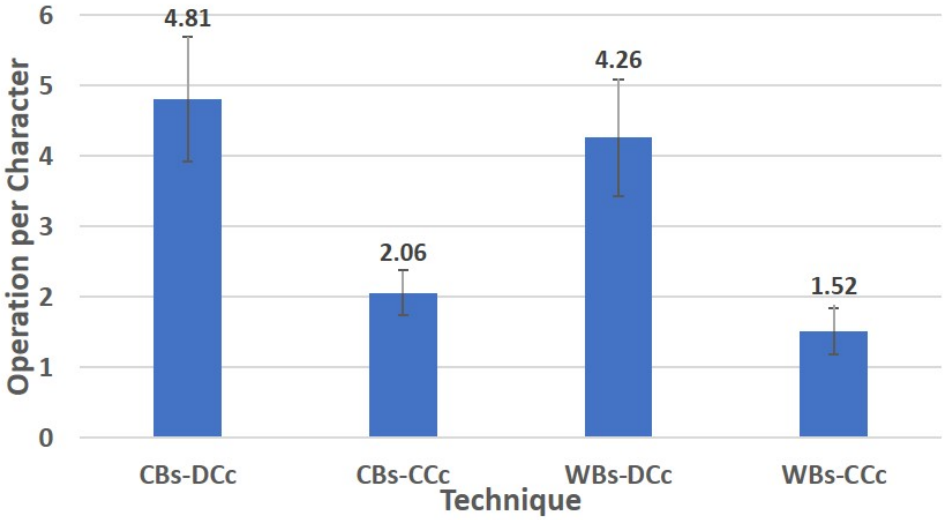

Fig. 4.1 Operation per character for four VR text revision techniques.

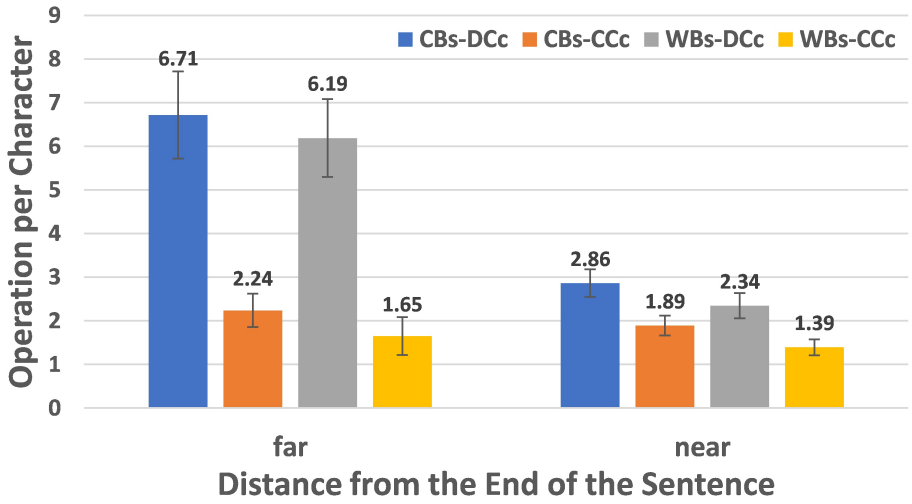

Fig. 4.2 Operation per character for targets far from or near the end of the sentence.

Correction time is used to describe the average duration needed to revise the given target in the sentence. In detail, it is the duration between the beginning of the trial and the moment when participants successfully submit the revised sentence in the system. Fig. 4.3 shows the statistical results of correction time for four proposed text revision techniques. Results showed that WBs-CCc required the shortest time (with the mean of 7978.19) to revise the target, while CBs-DCc cost the longest (with the mean of 9269.94). We found a significant effect of technique on the correction time with a Fried-

man test ($\chi^2(3) = 335.9$, p <.001). We also ran a post-hoc test and found significant differences among all technique pairs (all p <.001). Fig. 4.4 shows the correction time when participants revised targets with different distances. Results showed that revising targets that are far from the end of the sentence would spend more time. Meanwhile, when revising targets far away, techniques with continuous caret (cursor) control took less time to finish the revision.

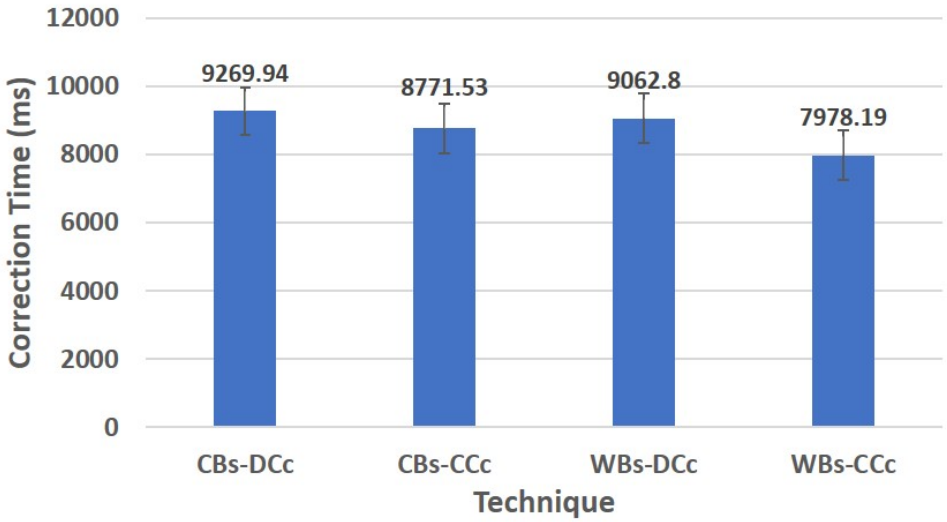

Fig. 4.3 Correction time for four VR text revision techniques.

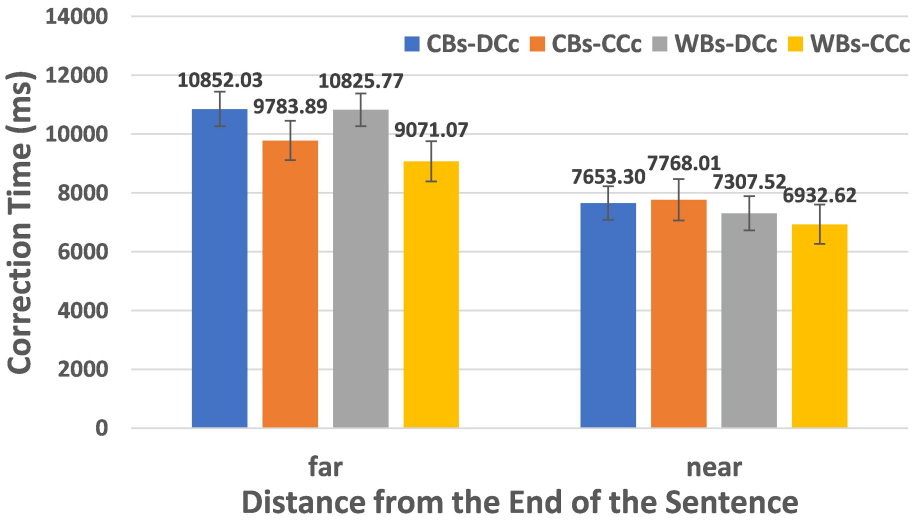

Fig. 4.4 Correction time for targets far from or near the end of the sentence.

We also count the average number of using the backspace (see Fig. 4.5) and caret (cursor) control (see Fig. 4.6) during the revision process. Results showed that the word-level backspace and continuous caret (cursor) control can significantly decrease the frequency of using the backspace and navigating the caret (cursor). WBs-CCc got the least number of backspace (with the mean of 1.04) and CBs-CCc got the least number of caret (cursor) control (with the mean of 2.32). We found a significant effect of technique on the number of backspace with a Friedman test ($\chi^2(3) = 252.08$, $p < .001$). For between two CBs-based techniques and between two WBs-based techniques, we did not find any significant differences with the post-hoc test. For caret (cursor) control, we found a significant effect of technique on the number of caret (cursor) control with a Friedman test ($\chi^2(3) = 1013.75$, $p < .001$). For between two DCc-based techniques and between two CCc-based techniques, we did not find any significant differences with the post-hoc test. Fig. and Fig. shows the number of backspace (known as "backspace frequency" in the figure) and caret (cursor) control (known as "caret (cursor) frequency" in the figure) when revising targets with different distance from the end of the sentence.

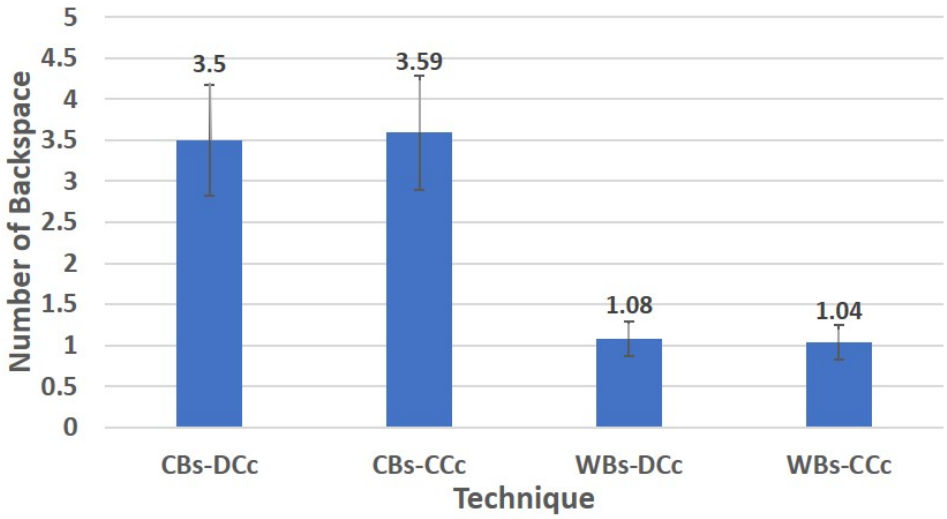

Fig. 4.5 Number of backspace for four VR text revision techniques.

Caret (cursor) control time is calculated as the total time consumption of navigating

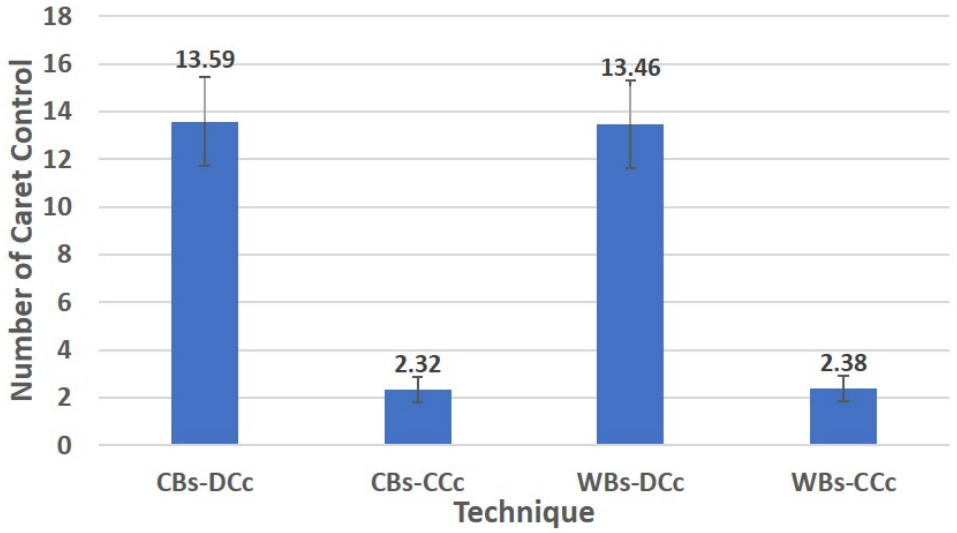

Fig. 4.6 Number of caret (cursor) control for four VR text revision techniques.

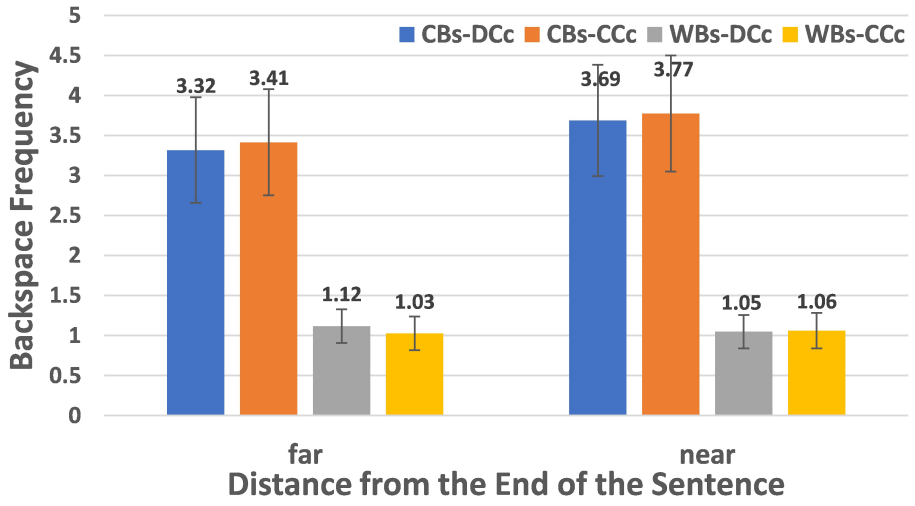

Fig. 4.7 Backspace frequency for targets far from or near the end of the sentence.

the caret (cursor) in each of the revision trial. Fig. 4.9 shows the statistical results of caret (cursor) control time for four proposed text revision techniques. Results showed that CCc-based techniques had better performance in caret (cursor) control time than DCc-based techniques. We found a significant effect of technique on caret (cursor) control time with a Friedman test ($\chi^2(3) = 755.29$, $p < .001$). Fig. 4.10 further shows that for targets no matter the place, techniques with the continuous caret (cursor)

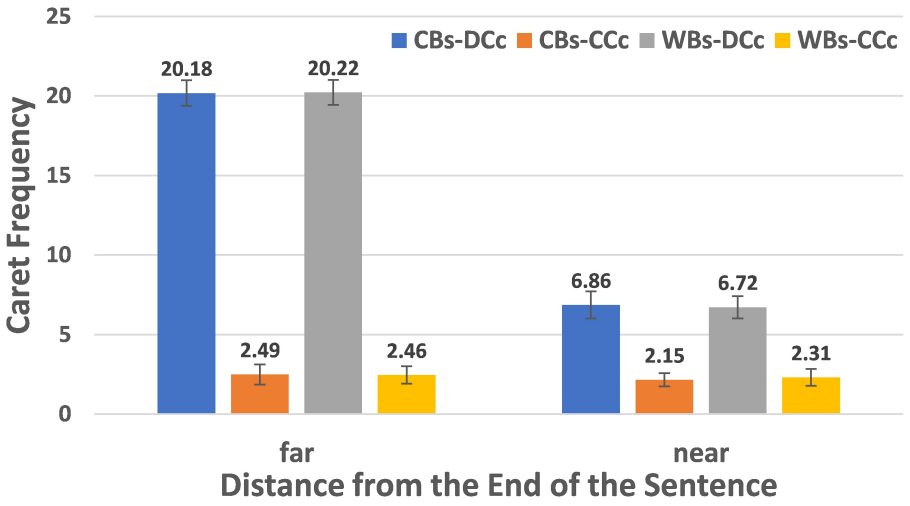

Fig. 4.8 Cursor frequency for targets far from or near the end of the sentence.

control can decrease the time consumption during the revision process.

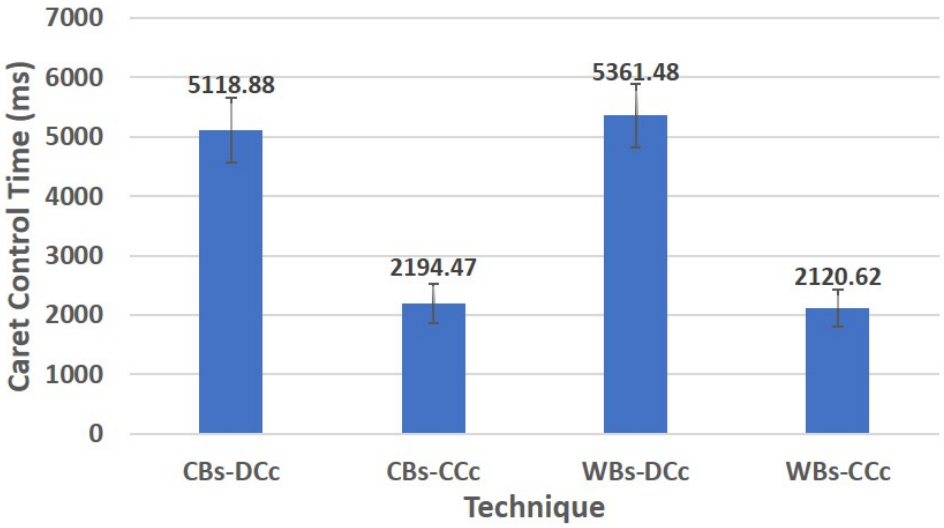

Fig. 4.9 Caret (cursor) control time for four VR text revision techniques.

Backspace time describes the average time consumption of pressing the backspace key when revising each target. It is calculated as the period between the previous button pressing and the selection moment of the backspace key. Fig. 4.11 shows the statistical results of backspace time for four proposed text revision techniques. Results showed that techniques with word-level backspace required less time to use the backspace than those

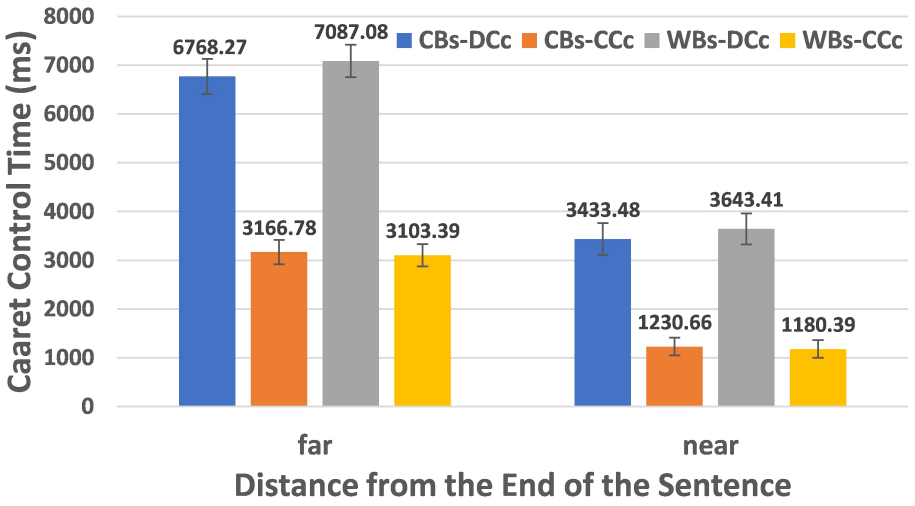

Fig. 4.10 Caret (cursor) control time for targets far from or near the end of the sentence.

techniques with character-level backspace. We found a significant effect of technique on caret (cursor) control time with a Friedman test ($\chi^2(3) = 196.14$, $p < .001$). Fig. 4.12 shows that when revising targets far from the end of the sentence, WBs-CCc required the least time to use backspace to finish the revision.

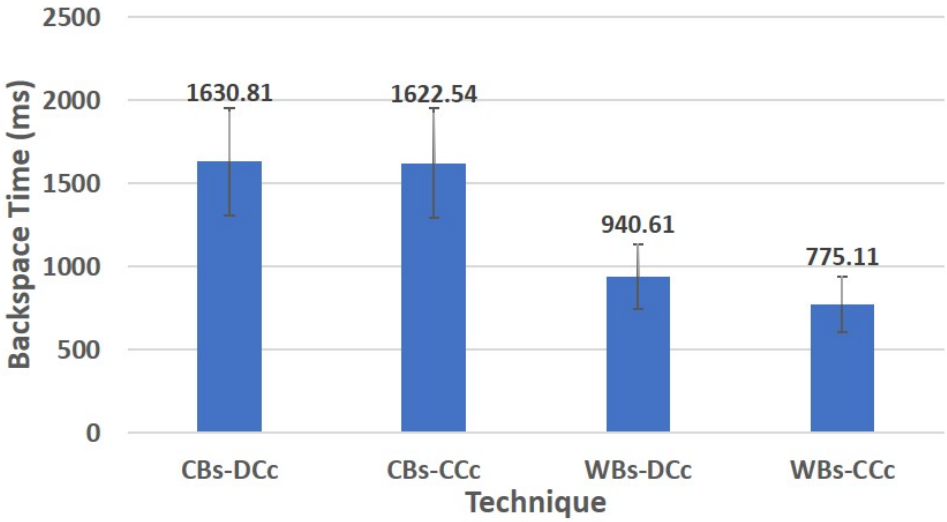

Fig. 4.11 Backspace time for four VR text revision techniques.

We evaluated participants' preference towards the four proposed techniques via NASA-TLX [33] (see Fig. 4.13) and SUS score (see Fig. 4.14). Overall, WBs-CCc

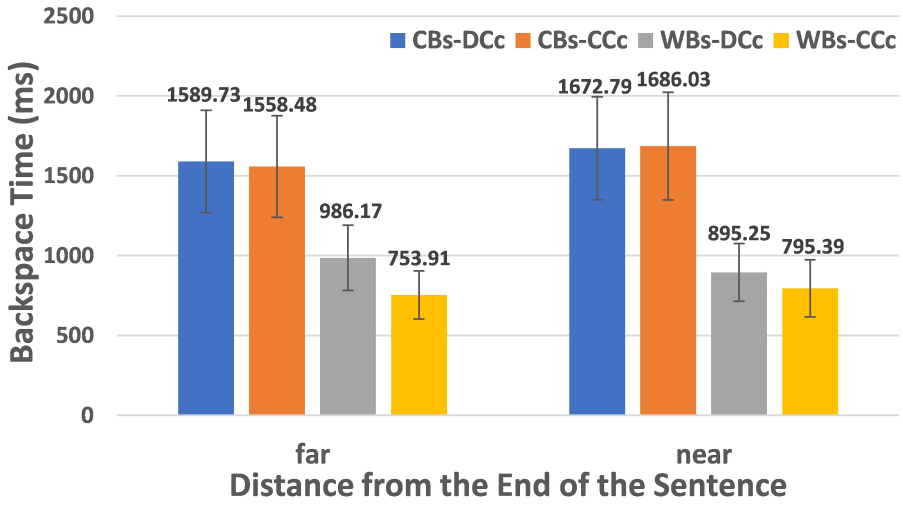

Fig. 4.12 Backspace time for targets far from or near the end of the sentence.

got the lowest NASA-TLX score (the lower the better) and the highest SUS score (the higher the better) among four techniques and among all participants. Additionally, in the NASA-TLX, all participants rated the most important factors when designing text revision techniques in VR, Effort, Performance, and Physical Demand are three most important factors that participants rated.

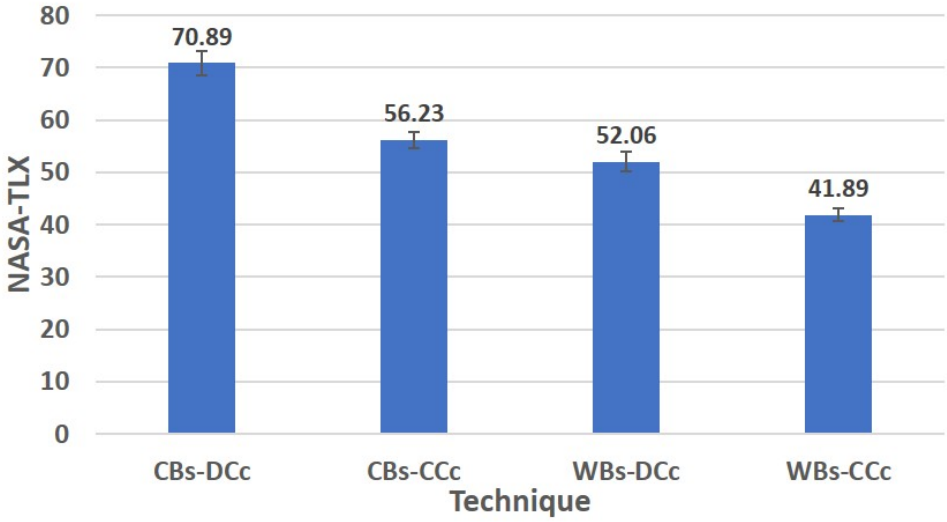

Fig. 4.13 NASA-TLX results for four VR text revision techniques.

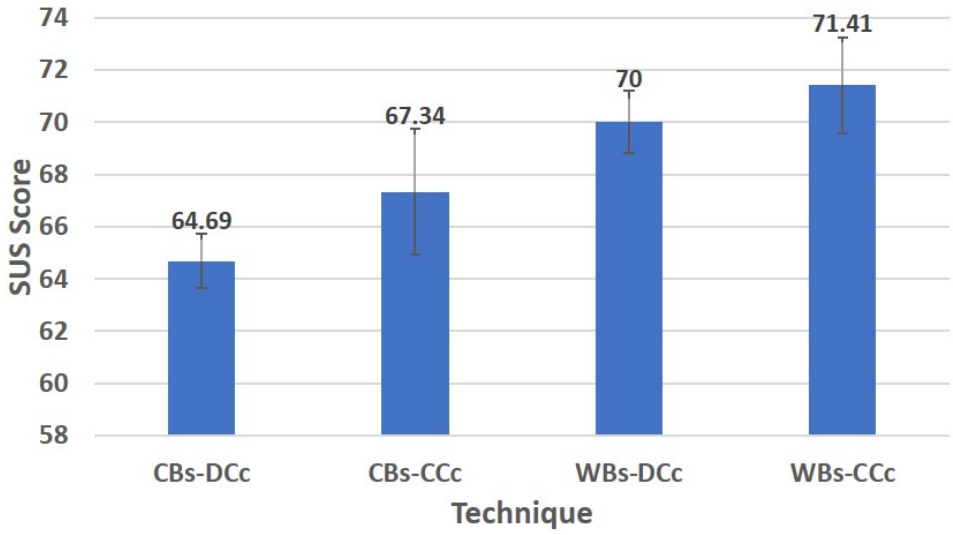

Fig. 4.14 SUS scores for four VR text revision techniques.

Chapter 5

Discussion

In this chapter, we focus on the following perspectives: 1) we discuss the scalability of our proposed design space, 2) we further analyze the results (for both quantitative and qualitative) got from the comparative user study, and 3) we will point out the limitation and future work for extensively research on text revision in VR applications.

5.1 Scability of the Design Space

In this study, we only include backsapce and caret (cursor) as two factors into the design space. In other words, we built this design space from the level of revision tools. We did this because backsapce and caret (cursor) are two obivious and avaliable factors that come into our mind when proposing the design space. We admit that the current design space is a simplified version for roughly summarizing current text revision possibilities.

We should also be aware of the fact that compared with text revision research on smartphones, VR text revision is more sophisticated with more factors involved during the design process. In detail, smartphone text revision has the inherent limitation of device, which we can focus more on exploring the text revision possibilities on a certain type of devices. However, for VR text revision, device is also a vital factor to consider. We cannot achieve a unified text revision solution that can be well-applied for various devices. For instance, WBs-CCc is currently the best option for text revision with

handheld controllers, but its performance may cannot be guaranteed when applied for conditions with bare hands or styli.

5.2 Backspace and Cursor during Revision

Results showed that word-level backspace reduced the frequency and time consumption of the backspace key. Compared with character-level backspace, users just need to perform one time of word-level backspace to achieve the similar performance of repetitive pressing. In our text revision design, we did not include the recovery mechanism in the system (e.g., as the Ctrl+Z function or the undo button). This decision brings potential risks to participants during the revision process. In detail, if participants mis-trigger the word-level backspace or press it multiple times. In that case, more attention and efforts will be needed to 1) check the whole sentence to confirm which word(s) has been mis-deleted and 2) reenter the mis-deleted words and proofread the sentence to make sure the recovery is qualified. Participants commented that, if there is no recovery mechanism, the extra efforts of dealing with the mis-operation would counteract the benefit brought from the word-level backspace.

Meanwhile, we keep the backspace available on the virtual keyboard to see how participants behave when they encounter typos during typing. Based on the log records and onsite observation, it is interesting to see that participants tend to use the backspace on the virtual keyboard (even though we already instructed them to only use the physical button on the handheld controller). We further asked participants about this behavior after the experiment. Participants gave the reasons and scenarios of using the backspace on the virtual keyboard. After summarizing their comments, we found that participants had a quick balance protocol when deciding the use of different backspaces. In general, it depends on the timing of observing the typo character: if that is just 1-2 characters

away from the current caret (cursor) position, using the backspace key on the keyboard would be the best choice; if the typo character has been already missed with multiple characters after it, using the word-level backspace would feed the need (in this condition, the cost of re-entering the correct characters would be regarded as necessary). Based on the findings above, we summarize the design suggestions for the use of backspace: 1) proper undo mechanisms should be involved in the VR typing system if the word-level backspace is provided to users, and 2) character-level backspace is always needed for quick deletion.

We further did a informal comparison about the preference of backspace on the handheld controller and that on the virtual keyboard. After the formal experiment, we asked five participants to revise another 30 sentences only with the virtual keyboard. After the revision, we asked their preferences and found that the backspace on the controller is more popular. The possible reason may be that 1) pressing the backspace on the controller can avoid the long-distance movement (compared with subtle finger movement) when navigating the virtual mallet to the backspace key on the virtual keyboard. 2) for repetitive backspace pressing, using the backspace on the virtual keyboard would bring repetitive acceleration changes during pressings, which may cause extra fatigue to participants.

We are shocked by the fact that the cursor and caret (cursor) control mechanism are missed in current VR text entry designs and systems. They, by default, set the caret (cursor) at the end of the input string and ban the movement flexibility of going between characters. Although this setting saves the space and effort of implementing caret (cursor) control methods (e.g., arrow keys), it makes text revision time-consuming or even impossible (for those designs without a backspace). With essential caret (cursor) control, users can navigate the caret (cursor) near the intended position and finish the revision with the backspace. This can significantly decrease the effort during revision

compared with that with backspace only. Results (Fig. 4.2 and Fig. 4.8) validated that participants with the continuous cursor control can decrease the number of operations. The performance is more obvious for the condition where participants were required to revise targets that are far from the end of the sentence.

However, similar to the backspace, caret (cursor) control may also cause negative effects on the text revision performance. During the design process, we often consider the target users as "perfect" individuals (making no unintended interactions as imagined). However, in the practical use, we found that participants sometimes cannot control the cursor well. Especially for the continuous caret (cursor) control, Fig. 4.8 showed that users may need extra attempts to navigate the caret (cursor) to the intended position after their first attempt. The possible reason is that, for continuous caret (cursor) control, the cursor moves quickly when sliding on the touchpad, participants need to focus and trace the cursor movement. However, there exists a small period of time offset between the moment when participants realized the cursor reached the position and the moment of executing the operation of lifting up the finger from the touchpad. This offset may cause the cursor finally stopped before or after the intended position. Additionally, when lifting the finger from the touchpad, there also exists a small-scale offset movement of the fingertip, which may also lead the unintended cursor movement.

According to the results of NASA-TLX questionnaire, we confirmed Effort, Performance, and Physical demand as three most important factors that users paid attention to when they evaluate text revision techniques in VR. This finding inspires us to summarize the design guideline from the user perspective for designers that, the novel proposed VR text revision techniques should show the power of dealing with exception conditions during the revision with less effort. Meanwhile, the techniques should make sure the revision can be properly and successfully finished with less physical (and mental) workload.

5.3 Design suggestions for VR Revision

The conducted studies and results not only shed lights on the essential understanding and summary of the research status of current text entry and text revision in VR but also contribute to the community of HCI, especially for practitioners and designers with valuable reference and suggestions when they attempt to propose useful and efficient VR text entry methods with enough consideration of text revision. Here we list the general design strategies concluded from the previous studies of this thesis.

First, from the motivation level of designing VR text entry techniques, designers should put make sure the proposed technique has at least the essential capability of handling various conditions during text entry and text editing rather than just chase for faster typing speed.

Second, for VR text revision enhancement, the proposed designs should focus on the ease of use, physical (and mental) workload, and the robustness of dealing with different revision conditions.

Third, the text revision design should involve both character-level deletion and word-level deletion mechanisms to satisfy different requirements of quick correction and word-level revision. Undo mechanisms are also needed to make sure users can have the chance to recover from their mis-operations.

Fourth, cursor control is a vital component that cannot be removed or ignored. Proper assistant and corrective algorithms should also be involved to make sure the accuracy of manipulation according to the characteristics of various devices.

Last but not least, when implementing the backspace and cursor control (two of the most frequently used tools), essential optimizations are required to ensure users will not encounter heavy physical fatigue when repetitively using those two tools.

5.4 Limitation and Future Work

This work is the initial step of revealing the lack concern of text revision in VR and providing the basic technique options to enhance the text revision performance in VR environments based on the combination use of backsapce and caret. There is still a long way to go for indepth research of VR text revision facilitation. In future work, we will first conduct a long-term field study to further investigate how users use our proposed text revision techniques in different real-life situations. Then, towards various input devices in VR, we will optimize the proposed techniques to fit the features of various devices to achieve the similar VR text revision performance. Finally, we will further investigate the performance of our proposed techniques when users revise text content in different conditions (e.g., during the game with time limit, sitting, or standing) with a series of comparative studies.

Chapter 6

Conclusion

Current developing VR techniques show us the promising future of transplanting daily activities into the virtual world. For the virtualization of office work, practical and efficient VR text input technique is one of the most vital and essential basis for both providing immersive interaction experience and maintaining the working productivity, especially for text-related tasks. Currently, numerous research and proposed techniques mainly contribute to the facilitation of typing speed, accuracy, and error correction features during typing in VR. Based on those improvements, they remarked themselves "useful and efficient" for VR text-related tasks. We found that previous research and methods cannot be qualified in the practical use because they only address the issue of entering the text into VR systems. However, there is a lack of essential consideration on the text manipulations after entering the text content, especially for text revision, the most common and representative text editing activity. Through our literature review on existing VR text entry techniques, we found that few research mentioned text revision in their research and thus no further discussions on the facilitation of VR text revision. Especially for VR typing with virtual keyboards, only backspace is available without discussing the effectiveness.

To fill this gap, we combine the use of backspace and cursor into our considerations for the following design of VR text revision techniques. Then, we confirmed the optional functions (for backspace and cursor) and further designed and validated four text revision techniques based on the combinational use of backspace and caret (cursor) in the

context of using virtual keyboards and handheld controllers. Results showed that using word-level backspace and continuous cursor control is capable of text revision in VR. This work and related result can be used as the groundwork and reference for the future design of VR text revision techniques. Our research also contribute to the community of VR text entry by reminding and raising the awareness and interests of text revision improvement when proposing text entry techniques.

Acknowledgement

I would like to express my deep and sincere gratitude to my research supervisor, **Prof. Xiangshi Ren**, for giving me the opportunity to do research, creating a great research environment in Human-Engaged Computing Laboratory. I would also truly thank him for his invaluable guidance throughout this research.

I am especially indebted to the committee members, **Prof. Shinomori Keizo** and **Prof. Shigemasu Hiroaki** for all of their guidance through this process; your discussion, ideas, and feedback have been absolutely invaluable.

My sincere gratitude also goes to my mentor, **Yang Li** for consistent support and guidance throughout this research. He supported me throughout this process and has constantly encouraged me when the tasks seemed arduous and insurmountable.

I would like to say a special thank you to **Xinhui Jiang** and **Chen Wang** for their assistance and encouragement and thank you to all the **CHEC and Ren Lab members** for their energy, understanding and help throughout my project.

I also would like to thank **Chiyo Kawagoe** for her help at the School of Information.

I would like to thank you to all those who participated in our experiment.

I am extremely grateful to my parents, **Mushun Zheng** and **Chanshan Wu**, for their love, prayers, caring, and sacrifices for educating and preparing me for my future.

Also, I would like to say special thank you to all parties who have helped this final project, but may not be realized by the author.

Last but not least, from the bottom of my heart I would like to express my gratitude for **Kochi University of Technology**, for giving me the opportunity to research and pursue my master degree.

References

[1] Ahmed Sabbir Arif et al. "Evaluation of a smart-restorable backspace technique to facilitate text entry error correction". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 5151–5162.

[2] Michael Bohan et al. "A psychophysical comparison of two stylus-driven soft keyboards". In: *Graphics Interface*. Citeseer. 1999, pp. 92–97.

[3] Costas Boletsis and Stian Kongsvik. "Controller-based text-input techniques for virtual reality: an empirical comparison". In: *International Journal of Virtual Reality (IJVR)* 19.3 (2019).

[4] Sabah Boustila et al. "Text typing in VR using smartphones touchscreen and HMD". In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 860–861.

[5] Sidney Bovet et al. "Using traditional keyboards in vr: Steamvr developer kit and pilot game user study". In: *2018 IEEE Games, Entertainment, Media Conference (GEM)*. IEEE. 2018, pp. 1–9.

[6] Doug A Bowman, Vinh Q Ly, and Joshua M Campbell. *Pinch keyboard: Natural text input for immersive virtual environments*. Tech. rep. Department of Computer Science, Virginia Polytechnic Institute & State ⋯, 2001.

[7] Doug A Bowman, Christopher J Rhoton, and Marcio S Pinho. "Text input techniques for immersive virtual environments: An empirical comparison". In: *Proceedings of the human factors and ergonomics society annual meeting*. Vol. 46. 26. SAGE Publications Sage CA: Los Angeles, CA. 2002, pp. 2154–2158.

[8] Thenille Braun Janzen et al. “Timing skills and expertise: discrete and continuous timed movements among musicians and athletes”. In: *Frontiers in psychology* 5 (2014), p. 1482.

[9] John Brooke. “SUS: a retrospective”. In: *Journal of usability studies* 8.2 (2013), pp. 29–40.

[10] Damien Brun, Charles Gouin-Vallerand, and Sébastien George. “Keycube is a Kind of Keyboard (k3)”. In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–4.

[11] Sibo Chen et al. “Exploring Word-gesture Text Entry Techniques in Virtual Reality”. In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–6.

[12] Daewoong Choi et al. “Designing hand pose aware virtual keyboard with hand drift tolerance”. In: *IEEE Access* 7 (2019), pp. 96035–96047.

[13] Wikimedia Commons. *KB United States Dvorak.* Website. `https://commons.wikimedia.org/wiki/File:KB_United_States_Dvorak.svg`. 2021.

[14] Wikimedia Commons. *QWERTY Keyboard.* Website. `https://commons.wikimedia.org/wiki/File:KB_United_States.svg`. 2018.

[15] Cutiekeys. *NormalVR/CutieKeys.* Website. `https://github.com/NormalVR/CutieKeys/`. 2017.

[16] Samir Dash. *BlueTap - The Ultimate Virtual-Reality (VR) Keyboard. Medium.* Website. `https://medium.com/eunoia-i-o/bluetap-the-ultimate-virtual-reality-vr-keyboard-77f1e3d57d6f`. 2017.

[17] Alan Dix et al. “Human-computer interaction”. In: *Harlow ua* (2000).

[18] Tafadzwa Joseph Dube and Ahmed Sabbir Arif. “Text entry in virtual reality: A comprehensive review of the literature”. In: *International Conference on Human-Computer Interaction*. Springer. 2019, pp. 419–437.

[19] John Dudley et al. “Performance envelopes of virtual keyboard text input strategies in virtual reality”. In: *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2019, pp. 289–300.

[20] Neil RN Enns and I Scott MacKenzie. “Touchpad-based remote control devices”. In: *CHI 98 conference summary on Human factors in computing systems*. 1998, pp. 229–230.

[21] Jacqui Fashimpaur, Kenrick Kin, and Matt Longest. “PinchType: Text Entry for Virtual and Augmented Reality Using Comfortable Thumb to Fingertip Pinches”. In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–7.

[22] Donald R Gentner et al. “A glossary of terms including a classification of typing errors”. In: *Cognitive aspects of skilled typewriting*. Springer, 1983, pp. 39–43.

[23] Gabriel González et al. “Evaluation of text input techniques in immersive virtual environments”. In: *New Trends on Human–Computer Interaction*. Springer, 2009, pp. 109–118.

[24] GoogleDaydream. *Daydream Labs: exploring and sharing VR’s possibilities*. Website. `https://blog.google/products/daydream/daydream-labs-exploring-and-sharing-vrs`. 2016.

[25] Daniel Gopher and David Raij. “Typing with a two-hand chord keyboard: will the QWERTY become obsolete?” In: *IEEE Transactions on Systems, Man, and Cybernetics* 18.4 (1988), pp. 601–609.

[26] Jens Grubert et al. “Effects of hand representations for typing in virtual reality”. In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2018, pp. 151–158.

[27] Jens Grubert et al. “Text entry in immersive head-mounted display-based virtual reality using standard keyboards”. In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2018, pp. 159–166.

[28] Jens Grubert et al. “The office of the future: Virtual, portable, and global”. In: *IEEE computer graphics and applications* 38.6 (2018), pp. 125–133.

[29] Jan Gugenheimer et al. “Facetouch: Enabling touch interaction in display fixed uis for mobile virtual reality”. In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 2016, pp. 49–60.

[30] Jie Guo et al. “Evaluation of Maslows Hierarchy of Needs on Long-Term Use of HMDs–A Case Study of Office Environment”. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 948–949.

[31] Aakar Gupta et al. “Rotoswype: Word-gesture typing using a ring”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–12.

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

[33] Sandra G Hart and Lowell E Staveland. “Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research”. In: *Advances in psychology*. Vol. 52. Elsevier, 1988, pp. 139–183.

[34] Morton L Heilig. *Sensorama simulator*. US Patent 3,050,870. Aug. 1962.

[35] Adrian H Hoppe et al. “qvrty: Virtual keyboard with a haptic, real-world representation”. In: *International Conference on Human-Computer Interaction*. Springer. 2018, pp. 266–272.

[36] Akira Ishii et al. “FistPointer: target selection technique using mid-air interaction for mobile VR environment”. In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2017, pp. 474–474.

[37] Haiyan Jiang and Dongdong Weng. “HiPad: Text entry for Head-Mounted Displays Using Circular Touchpad”. In: *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2020, pp. 692–703.

[38] Youngwon R Kim and Gerard J Kim. “Hovr-type: Smartphone as a typing interface in vr using hovering”. In: *2017 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE. 2017, pp. 200–203.

[39] Pascal Knierim et al. “Opportunities and Challenges of Text Input in Portable Virtual Reality”. In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–8.

[40] Pascal Knierim et al. “Physical keyboards in virtual reality: Analysis of typing performance and effects of avatar hands”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–9.

[41] Andreas Komninos et al. “A glimpse of mobile text entry errors and corrective behaviour in the wild”. In: *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. 2018, pp. 221–228.

[42] Ben Kuchera. *Google reports that drums make a great virtual reality keyboard.* Website. `https://www.polygon.com/2016/5/19/11715654/google-daydream-vr-drums`. 2016.

[43] Falko Kuester et al. “Towards keyboard independent touch typing in VR”. In: *Proceedings of the ACM symposium on Virtual reality software and technology*. 2005, pp. 86–95.

[44] Vladimir Iosifovich Levenshtein. “Binary codes capable of correcting deletions, insertions and reversals.” In: *Soviet Physics Doklady* 10.8 (Feb. 1966). Doklady Akademii Nauk SSSR, V163 No4 845-848 1965, pp. 707–710.

[45] Yang Li et al. “Swap: A Replacement-based Text Revision Technique for Mobile Devices”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–12.

[46] Stan Liebowitz and Stephen E Margolis. “Policy and path dependence: from QWERTY to Windows 95”. In: *Regulation* 18 (1995), p. 33.

[47] Stan J Liebowitz and Stephen E Margolis. “The fable of the keys”. In: *The Journal of Law and Economics* 33.1 (1990), pp. 1–25.

[48] Jia-Wei Lin et al. “Visualizing the keyboard in virtual reality for enhancing immersive experience”. In: *ACM SIGGRAPH 2017 Posters*. 2017, pp. 1–2.

[49] Xueshi Lu et al. “Depthtext: Leveraging head movements towards the depth dimension for hands-free text entry in mobile virtual reality systems”. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 1060–1061.

[50] Cuixia Ma et al. “An adaptive sketching user interface for education system in virtual reality”. In: *2009 IEEE International Symposium on IT in Medicine & Education*. Vol. 1. IEEE. 2009, pp. 796–802.

[51] I Scott MacKenzie and R William Soukoreff. “Phrase sets for evaluating text entry techniques”. In: *CHI’03 extended abstracts on Human factors in computing systems*. 2003, pp. 754–755.

[52] Mark McGill et al. “A dose of reality: Overcoming usability challenges in vr head-mounted displays”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pp. 2143–2152.

[53] Carsten Mehring et al. “KITTY: Keyboard independent touch typing in VR”. In: *IEEE Virtual Reality 2004*. IEEE. 2004, pp. 243–244.

[54] Tim Menzner et al. “A capacitive-sensing physical keyboard for vr text entry”. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 1080–1081.

[55] Kyungha Min. “Text input tool for immersive VR based on 3× 3 screen cells”. In: *International Conference on Hybrid Information Technology*. Springer. 2011, pp. 778–786.

[56] Jan Noyes. “The QWERTY keyboard: A review”. In: *International Journal of Man-Machine Studies* 18.3 (1983), pp. 265–281.

[57] Taihei Ogitani et al. “Space saving text input method for head mounted display with virtual 12-key keyboard”. In: *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*. IEEE. 2018, pp. 342–349.

[58] Alexander Otte et al. “Towards utilizing touch-sensitive physical keyboards for text entry in virtual reality”. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 1729–1732.

[59] Duc-Minh Pham and Wolfgang Stuerzlinger. “Hawkey: Efficient and versatile text entry for virtual reality”. In: *25th ACM Symposium on Virtual Reality Software and Technology*. 2019, pp. 1–11.

[60] Manuel Prätorius et al. “Sensing thumb-to-finger taps for symbolic input in vr/ar environments”. In: *IEEE computer graphics and applications* 35.5 (2015), pp. 42–54.

[61] Vijay Rajanna and John Paulin Hansen. “Gaze typing in virtual reality: impact of keyboard design, selection method, and motion”. In: *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. 2018, pp. 1–10.

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

[63] Robert Rosenberg. "Computing without mice and keyboards: text and graphic input devices for mobile computing". PhD thesis. UCL (University College London), 1998.

[64] Amal Sirisena. "Mobile text entry". In: (2002).

[65] Jeongmin Son et al. "Improving Two-Thumb Touchpad Typing in Virtual Reality". In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–6.

[66] Robert William Soukoreff. "Text entry for mobile systems: Models, measures, and analyses for text entry research." In: (2003).

[67] Marco Speicher et al. "Selection-based text entry in virtual reality". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–13.

[68] Keith Vertanen and Per Ola Kristensson. "A versatile dataset for text entry evaluations based on genuine mobile emails". In: *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. 2011, pp. 295–298.

[69] G. Walker. "A review of technologies for sensing contact location on the surface of a display". In: *Journal of The Society for Information Display* 20 (2012), pp. 413–440.

[70] James Walker et al. "Efficient typing on a visually occluded physical keyboard". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 5457–5461.

[71] Wikipedia contributors. *Backspace — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-January-2021]. 2020. URL: https://en.wikipedia.org/w/index.php?title=Backspace&oldid=996204475.

[72] Andrew D Wilson and Maneesh Agrawala. "Text entry using a dual joystick game controller". In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006, pp. 475–478.

[73] Chien-Min Wu et al. "A virtual reality keyboard with realistic haptic feedback in a fully immersive virtual environment". In: *Virtual Reality* 21.1 (2017), pp. 19–29.

[74] LI X X. *Virtual Reality Special Report.* Website. https://stock.tianyancha.com/qmp/report/2/13942fc810d68718897e9cc20c7a580c.pdf. 2019.

[75] Wenge Xu et al. "Ringtext: Dwell-free and hands-free text entry for mobile head-mounted displays using head motions". In: *IEEE transactions on visualization and computer graphics* 25.5 (2019), pp. 1991–2001.

[76] Naoki Yanagihara and Buntarou Shizuki. "Cubic keyboard for virtual reality". In: *Proceedings of the Symposium on Spatial User Interaction*. 2018, pp. 170–170.

[77] Chun Yu et al. "Tap, dwell or gesture? Exploring head-based text entry techniques for HMDs". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 4479–4488.

[78] Difeng Yu et al. "PizzaText: Text entry for virtual reality systems using dual thumbsticks". In: *IEEE transactions on visualization and computer graphics* 24.11 (2018), pp. 2927–2935.

[79] Zekromtor. *Curved Split Keyboard Prototype*. Website. https://deskthority.net/viewtopic.php?t=8472. 2014.

[80] Shumin Zhai, Michael Hunter, and Barton A Smith. "The metropolis keyboard-an exploration of quantitative techniques for virtual keyboard design". In: *Proceedings*

of the 13th annual ACM symposium on User interface software and technology. 2000, pp. 119–128.

[81] FengJun Zhang, GuoZhong Dai, and Xiaolan PENG. "A survey on human-computer interaction in virtual reality". In: *Scientia Sinica Informationis* 46.12 (2016), pp. 1711–1736.

[82] Mingrui Ray Zhang, He Wen, and Jacob O Wobbrock. "Type, then correct: Intelligent text correction techniques for mobile text entry using neural networks". In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 2019, pp. 843–855.

Appendix A

Example Sentences of the Experiment

Table A.1 Examples of revision targets in different types

Error type	Near error
Omission (near error)	Required sentence: i would expect an answer asap
	Current sentence: i would expect an asap
Omission (far error)	Required sentence: did you not read my first email
	Current sentence: did you read my first email
Insertion (near error)	Required sentence: i am out of town until friday
	Current sentence: i am out of town until next friday
Insertion (far error)	Required sentence: let me know if i can help
	Current sentence: let me know that if i can help
Substitution (near error)	Required sentence: today has been hard for me
	Current sentence: today has been hard with me
Substitution (far error)	Required sentence: much better than carrying a laptop
	Current sentence: much worse than carrying a laptop