LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-850222

Design and Evaluation of Alphabetic and Numeric InputMethods for Virtual Reality

D. Kutak, D. Langlois, R. Rozic, J. Byska, H. Miao, S. Kriglstein, B. Kozlikova

June 13, 2023

ELSEVIER Computer & Graphics

Disclaimer

Computers & Graphics (2025)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Design and evaluation of alphabetic and numeric input methods for virtual reality

Double-blind submission, David Kuťák[1,*], Danielle Langlois[2], Roman Rozič[3], Jan Byška[4], Haichao Miao[5], Simone Kriglstein[6], Barbora Kozlíková[7]

ARTICLE INFO

Article history:
Received July 28, 2025

Keywords: virtual reality, head-mounted displays, user evaluation, alphabetic input, numeric input, alphanumeric input, virtual keyboards, writing

ABSTRACT

In today's virtual reality (VR), users have various ways to influence their VR experience, including alphanumeric input. While typing characters and numbers is straightforward on desktop computers, it presents challenges and opportunities in head-mounted display VR due to specific interaction methods and a lack of real-world visual stimuli. Addressing these open questions, our work implements and evaluates ten approaches to alphabetic and numeric inputs in VR. We describe the design motivation behind these input methods and evaluate them in a user study with 40 participants divided into groups for alphabetic and numeric keyboards. This comparison investigates each method's performance and user interactions. Our findings suggest that different input methods significantly impact words per minute and error rates, and that certain keyboard designs may receive better subjective evaluations despite poorer objective performance.

1. Introduction

This paper focuses on alphabetic and numeric input in head-mounted display virtual reality (HMD VR), as many of the tasks performed in virtual environments relate to this activity. Writing can have many goals, involving commanding a software system, creating an article, communicating with friends, or annotating complex scientific data. It also takes various forms—it may include letters, numbers, and special characters, and all of them are given various meanings based on the chosen language. The importance of text entry in virtual reality was discussed, for example, by Fourrier et al. [1]. They focused on industrial data entry scenarios, which is an area requiring a great deal of textual input combining letters, symbols, numbers, and special characters. Text input is important also for the educational scenarios studied by Motejlek et al. [2]. They conclude that while the effectiveness of learning is higher when the students answer questions via direct textual input [3], transforming such systems to virtual reality is challenging due to the current input limitations.

Given the importance of writing, we present and evaluate a set of methods focused on alphabetic and numeric input in HMD VR. In our context, the term *alphabetic input* refers to input methods supporting and primarily targeting the writing of letters, while the term *numeric input* refers to methods designed mainly for inputting numerical data. As for methods that are *alphanumeric*, they often fall under the *alphabetic* category since their primary design motivation is typically textual input. The division into purely alphabetic and numeric inputs studied in this publication is due to our specific application domain—this research was initiated by a desire to see what type of input would be most suitable in the context of VR molecular visualization and modeling. This area exhibits various needs for

*Corresponding author:
e-mail: kutak@mail.muni.cz (David Kuťák)
[1]Masaryk University, Brno, Czech Republic, kutak@mail.muni.cz
[2]Masaryk University, Brno, Czech Republic, 528203@mail.muni.cz
[3]Masaryk University, Brno, Czech Republic, 485422@mail.muni.cz
[4]Masaryk University, Brno, Czech Republic, University of Bergen, Bergen, Norway, byska@mail.muni.cz
[5]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, California, United States, miao1@llnl.gov
[6]Masaryk University, Brno, Czech Republic, kriglstein@mail.muni.cz
[7]Masaryk University, Brno, Czech Republic, kozlikova@mail.muni.cz

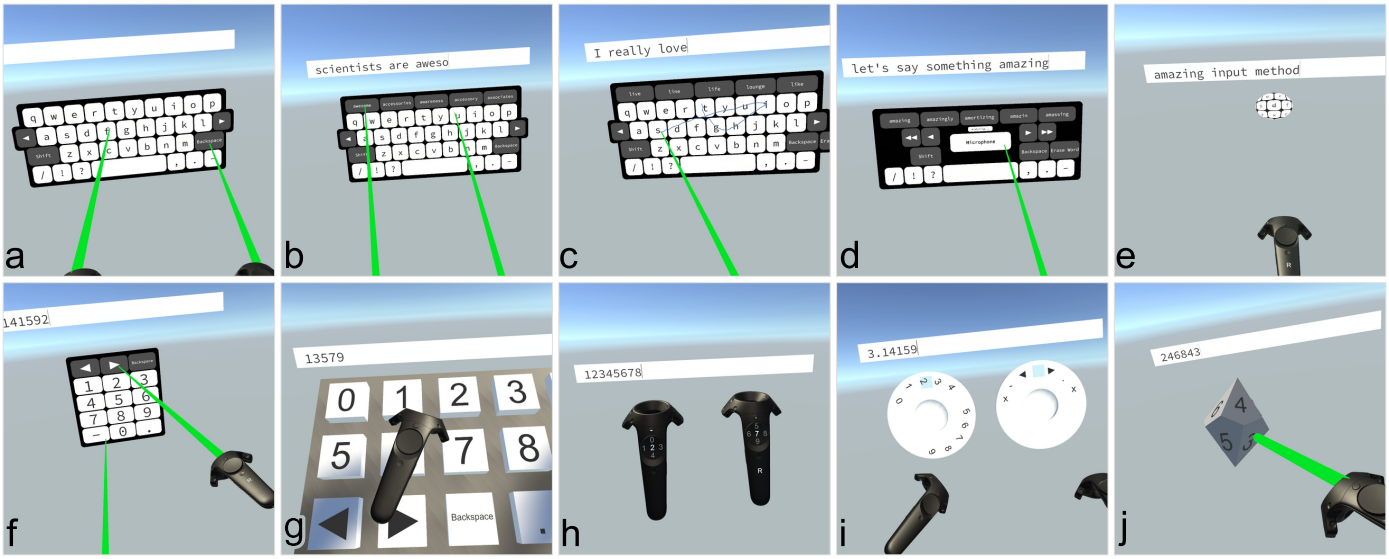

Fig. 1. An overview of all ten keyboards that were developed and evaluated in this study. The methods were separated into two categories based on the type of content produced, namely to alphabetic writing – a) to e) – and numeric writing – f) to j).

input of characters and numbers. For example, numeric input is typically sufficient when searching for particular amino acid residues in a protein chain or when visualizing parts of a DNA strand in chromatin complexes. Since this task often requires simultaneous input and visual exploration of the data, it is desired to obstruct the view of the structure as little as possible, thus the smaller the keyboard is, the better. On the other hand, when annotating molecular data or creating protein sequences, alphabetic input is needed. Therefore, our goal is to explore both worlds separately, which might potentially be followed by other researchers combining our input methods into unified alphanumeric techniques. In summary, the contributions of our research are:

- Implementation of ten methods for alphabetic and numeric input in HMD VR, including novel keyboards, methods inspired by existing research, as well as approaches adapted from other domains (see Section 3).
- Comparison-based evaluation of these input methods, focusing on their quantitative and qualitative characteristics. In particular, we aim to answer two main research questions:
  - **RQ1**: What are the best and worst performing methods in terms of writing speed and error rate, and what aspects mostly define their performance?
  - **RQ2**: How strongly does the users' subjective experience (i.e., personal keyboard preference) correspond to the objective data (i.e., measured keyboard performance)?

2. Related Work

This section categorizes the key existing works in this area, mostly focusing on approaches targeting HMD VR.

2.1. Alphabetic Input

Methods for alphabetic input are probably one of the most researched input modalities in the context of immersive virtual environments (IVEs), and thus they are discussed first.

2.1.1. Hardware Solutions

Although our goal was to use existing unaltered VR hardware instead of building a new one, there is a body of relevant work in this area worth mentioning. The authors of the so-called Vitty method augmented the capabilities of HTC Vive controllers by adding finger buttons to emulate QWERTY-style touch typing [4]. In the works of Menzner and Otte [5, 6], the traditional physical keyboard was extended with capacitive-sensing possibilities, enabling to detect not only press interaction but also touch. In FaceTouch, the touch screen was attached to the front part of the headset, enabling users to input text by using the attached display [7]. A similar idea was explored by Lee et al. [8], and Hutama et al. [9].

Apart from augmenting the existing hardware, completely novel interaction devices were also proposed. Son et al. [10] developed hand-held touchpad controllers to explore the challenge of two-thumb typing in VR—inspiring us to explore the two-thumb typing direction as well. In the work of Witt et al. [11], a sensor-mounted glove was proposed, enabling inputting text using a set of pre-defined gestures. In RotoSwype [12], a gesture-tracking ring was developed to input text using hand movements. Interesting hardware for detecting tapping was presented in TapID [13]—a wrist-worn sensor used with headset-based tracking, enabling to emulate touch screen-resembling writing experience. Finally, Keycube, presented by Brun et al. [14, 15], offers 80 keys placed on the sides of a hand-held cubic device, suitable for alphanumeric input, and various additional tasks. This concept also served as an inspiration for the design of some of our methods.

2.1.2. Virtual Keyboard Solutions

The so-called Drum-like keyboard was presented by Google's Daydream Labs and further evaluated by Boletsis et al. [16, 17, 18]. In the case of this keyboard, the controllers are used as drumsticks, whereas every key can be seen as a small drum. Two additional interesting concepts were presented by Yanagihara et al. The first one, the cubic keyboard, exploits the three-dimensionality of IVE by placing the keys in a 3D array with 3x3x3 elements [19]. The second concept also takes advantage of better spatial perception in HMD VR, yet this time it is realized by presenting the user with a curved keyboard [20]. Various techniques, ranging from the usage of controllers to head and hand tracking, were presented by Speicher et al. [21], with the raycasting approach performing best in the overall results. Several presented methods focus on the usage of the HTC Vive controller's trackpad as a means of textual input, inspiring us to follow this direction as well [22, 23, 24]. Tominaga et al. [25] investigated the effects of the virtual keyboard's position and angle on the typing performance, concluding that the tested angles did not influence typing speed, yet the position influenced the user's fatigue. Similarly, Dube et al. [26] explored the effect of different key shapes and dimensions on typing performance, concluding the square-shaped keys performed the best overall. Different key dimensions were also exploited in the work of Yang et al. [27], proposing spatial-enhanced virtual keyboards. In the work of Knierim et al. [28], various representations of an avatar's hand were compared, revealing that they do not have a significant influence on the performance of experienced typists. Similar results were reported by Grubert et al. [29]. Other interesting research results were described by Wan et al. [30], exploring methods for alphanumeric writing with a special focus on the input of special characters. They discovered that mapping the keyboard mode switching to controllers' buttons, instead of using virtual buttons corresponding to physical keyboard counterparts, may lead to increased performance. While most research focuses on text addition, Li et al. [31] went in another direction, studying how to improve the text revision via the combined use of backspace and caret control—their focus on revision inspired us to design unified caret control mechanisms.

2.1.3. Head- and Hand-Tracking Solutions

Apart from using the controllers, textual input can also be realized via hand-tracking approaches, potentially leading to more natural interaction with the virtual world. Ogitani et al. [32] presented a technique supporting the projection of the keyboard onto the palm of a user's hand. In the approaches by Adhikary and Vertanen [33], Tomaru and Tanaka [34], Jimenez and Schulze [35], and Pastor [36], hands of the user are used to select a key on a virtual representation of a classical keyboard layout. Compared to this, some textual input methods rely on hand gestures [37, 38, 39]. Similarly, head tracking might also be used as a means of alphabetic input [21, 40]. Yu et al. [41] explored three approaches to head-based text entry, finding out that the one based on a swipe-style keyboard performed surprisingly well in terms of writing speed. In the case of eye tracking approaches [42, 43], Rajanna et al. [42] investigated how the keyboard design, selection method, and motion in the field of view influence gaze typing performance.

2.1.4. Speech Recognition Solutions

While voice recognition in VR often serves primarily for controlling the system via spoken commands [44], approaches using speech as a form of textual input exist as well. For example, dedicated solutions like SWIFTER were developed, making the speech a core aspect of the text input technique [45]. SWIFTER augments the speech input with user-controlled error-correction techniques, enabling this method to remain simple from the interaction perspective, yet powerful enough to be practically usable. Derby et al. [46] studied the effect of a noisy environment on the speed and accuracy of Microsoft HoloLens dictation input, concluding that the accuracy started to be problematic when noise levels exceeded 60 dB.

2.1.5. Gesture-based Solutions

Word-gesture text entry was presented by Chen et al. [47], comparing two swipe-style typing keyboards interacted with using a controller and touch screen, with the controller version reaching better results. Similarly, Kern et al. [48] compared tap-based and swipe-based keyboards in the context of both virtual and augmented realities, obtaining results in favor of the tap-style approach. Another area being explored is handwriting-based textual input. Various approaches, such as neural networks, were employed to improve the recognition of written text [49, 50, 51]. Fourrier et al. [1] combined hand-tracking and hand-writing to propose an interaction technique closely resembling real-world writing experience. There were many additional interesting approaches in the world of controller-based input methods, focusing, for example, on the writing of Japanese characters, machine learning-based motion gesture detection, or 3DOF-only devices [52, 53, 54].

2.2. Numeric Input

The existing research targeting exclusively input of numerical data in HMD VR is rather small. In the work of Georgios Lepouras, several methods for numeric input are presented, involving the usage of gloves, wireless numeric keyboard, and gamepad [55, 56]. Among other outcomes, the results suggested that gesture-based interaction, e.g., using gloves, might not be the most suitable choice in the context of a numeric input scenario. In the work of Gao et al. [57], the tilt-click input technique was proposed. It was shown that such an approach might find use in small mobile devices. The eyes-free aspect of this keyboard inspired us when designing one of our numeric input methods.

2.3. Comparison Studies

Several of the existing works share a similar comparison-focused approach as we follow. Boletsis et al. [18] evaluated four keyboard designs, inspired by the already existing methods. Their study provides interesting observations we considered during our work. However, the limitations of their work include a focus strictly on alphabetic input and omitting the

auto-completion methods. Five text-entry methods were compared in the work of Speicher et al. [21]. Despite presenting various interesting outcomes across a wide design space, their work shares the same limitations as the work of Boletsis et al.

Lepouras [56], on the other hand, compares five methods for numeric input in VR. However, this work completely omits techniques designed for common VR controllers, preferring hand gestures or different types of hardware instead. The work of Wan et al. [30] also draws some similarities. However, their work focuses more on the aspect of switching between different keyboard modes, determining what characters are about to be written. Thus, this work complements our results in some aspects. Similarly, also the works of Derby et al. [46], Yang et al. [27], Dube et al. [26], and Li et al. [31] can be seen as complementary studies.

3. Evaluated Keyboards

In our work, we are targeting several challenges that were not yet tackled in the existing studies. However, some of them were stated as a future avenue of research in publications. Apart from being inspired by the existing methods, we explore the auto-completion functionality—suggested future direction in Boletsis et al. [18]. Furthermore, we investigate and evaluate a method that combines word-level and character-level input, as proposed by Yu et al. [41] in their future research directions. Finally, we also focus on the evaluation of purely numerical methods that are very scarcely researched so far.

We split the evaluated methods into purely alphabetic and purely numeric since both types of input have their place in VR HMDs, and their different domains enable us to design methods combining various interaction paradigms and design goals. For example, some of the keyboards rely on raycasting-based controls, as this technique is ubiquitous and well-performing [21], others on direct manipulation. Similarly, some methods are unimanual while others are bimanual by design-—and in some cases, this aspect is entirely up to the user. An important part of textual input is the ability to correct errors. In this area, our methods employ character-level backspace granularity and discrete caret control continuity [31].

Our keyboards are specifically designed for compatibility with standard HTC Vive VR controllers. While alternative interaction methods, such as hand-tracking, were considered, we prioritize controller-based techniques for two reasons. Firstly, the majority of VR headsets come bundled with controllers, which are readily available for interaction within the virtual environment. This contrasts with hand-tracking, which, although increasingly integrated, often requires additional hardware for support on some VR headsets. Secondly, VR controllers offer a unique set of features, including precise positional tracking, multiple buttons, and haptic feedback. Replicating these features with alternative methods may prove challenging. Therefore, if VR applications benefit from the capabilities of VR controllers, it is preferable to utilize them also for textual interactions rather than adopting a different interaction methodology for this particular task.

3.1. Alphabetic Input

We implemented and then evaluated five alphabetic keyboards, each of them optimized for writing common English sentences. We decided to explore three core aspects in this area. First, since the classical two-dimensional QWERTY keyboard often remains a natural first choice, thanks to its well-known layout, simple interactions, and straightforward integration to a larger 2D interface, we decided to explore this type of keyboard and its enhancements to better understand its capabilities. Then, we focused on the use of voice, enabling us to see the real-world performance of speech-to-text technology with non-native speakers. Finally, since VR keyboards are often rather large, they may occupy a significant amount of the user's view, potentially obstructing some important information. Therefore, we came up with a keyboard designed to tackle this challenge.

Raycast alphabetic: The most basic alphabetic keyboard, shown in Figure 1 (a), utilizes only ray casting. In this case, the user can use any controller to write letters selected by a ray casted from the controller. Square-shaped keys were used as per the suggestion of Dube et al. [26]. Since this type of keyboard is common in the majority of VR applications, it serves as a baseline for other methods.

Raycast dictionary: Similar to many current smartphone keyboards, this method (Figure 1 (b)) extends the basic raycasting keyboard with a row of text suggestions, aiming to simplify the completion or correction of the current word. A comparable approach was used also by Chen et al. [47] in their Swipe keyboard. The implementation uses an underlying word dictionary—unigram dataset consisting of 150,000 entries derived from Google Web Trillion Word Corpus—and n-grams model to suggest the most suitable completion of the current word. To perform the auto-completion or correction operation, the user can click the desired word in the top row. The selected word is then automatically followed by a space, as expected according to the feedback from our initial pilot study. This keyboard was selected since it should improve the writing speed without influencing the writing interaction. Thus, the user may take advantage of text suggestions, but they should not obstruct the writing experience, leading to an improved user experience.

Swipe: This word-gesture text entry method, known from smartphones, as well as the work of Chen et al. [47], has the same layout as *Raycast dictionary* keyboard, including the presence of text suggestions. However, in this case, the primary means of writing is by drawing a line over letters forming the desired word (Figure 1 (c)). After the line is drawn, the system evaluates the closest matching word, using the Greedy Asymmetric Dynamic Time Warping algorithm [58], and outputs it into the text field. Following the suggestions of Yu et al. [41], our Swipe keyboard also supports character-level input performed without a drawing gesture. This is useful if the user wants to correct small errors or write words unknown to the underlying text engine. Thus, this method aims to provide gesture-like input interaction without hindering the experience if the primary input means cannot be used. The expected benefit of Swipe lies in its gesture-based approach, potentially requiring less precision and interaction overhead during writing.

Speech to Text: As the name suggests, this input method

converts spoken input into a textual outcome (Figure 1 (d)). It relies on Microsoft Azure platform [59] to perform the conversion from speech to text and supports the conversion of single words as well as whole sentences. A strong advantage of this approach is the usage of speech, which is a very natural mode of interaction. Additionally, the method conceptually does not necessarily require user interface elements. While the user can write only using voice, we were inspired by SWIFTER method [45] and added text suggestions to simplify corrections. We assume that this can be beneficial namely for non-native speakers whose pronunciation can often be misinterpreted by the translator. All in all, this keyboard was included as is strives for a strong performance while feeling very natural to use.

Zoom: This keyboard follows the same layout as the basic *Raycast alphabetic* keyboard, yet the user looks at it through a small lens (Figure 1 (e)). The writing works as follows—first, the user presses the dedicated controller button to align the keyboard with the center of the lens, i.e., defines the optimal position for comfortable writing. Then, pressing the controller's trigger button outputs the letter corresponding to the key selected by a cursor positioned in the center of the lens. This cursor slides on the keyboard as the controller moves. In the case of this method, we aimed to see the influence of reduced visibility on the measured performance and users' satisfaction since making the keyboard smaller has many benefits, yet they should not come at an unbearably high price of making the keyboard unusable.

3.2. Numeric Input

Similar to alphabetic methods, five numeric input methods were implemented and subsequently evaluated. While both types of input can be theoretically covered by techniques for general alphanumeric input, we have decided to separate them. First, there are situations where it is sufficient to input only numerical data, therefore, we wanted to explore the ways of doing so. Second, the input domain of numerical methods is smaller since 12 characters—including the minus sign and decimal separator—are sufficient to write any number. This poses fewer constraints on the interactions and enables the exploration of more varying interaction approaches. Thus, while we use these methods to write numbers, our discoveries might be applicable beyond this domain in other applications where the user may need to choose from a limited set of options. As for the keyboards themselves, we decided to include keyboards offering both well-known and novel layouts, as well as various kinds of interaction means.

Raycast numeric: This keyboard is a numeric counterpart of the *Raycast alphabetic* keyboard, also serving as a baseline (Figure 1 (f)). The ordering of numbers corresponds to the well-known layout of telephone keypads, defined by the ITU-T E.161 standard [60].

Buttons: In this keyboard, the user is presented with a set of virtual buttons that imitate real-world pressing interaction, exploiting the natural way buttons are typically interacted with (Figure 1 (g)). The buttons are square-shaped and ordered in three rows to keep the keyboard reasonably compact yet still precise in selecting keys. By including this keyboard, we aim to study the impact of direct manipulation on typing performance and user satisfaction.

Trackpad: This input method (Figure 1 (h)) relies on two-thumbs typing methodology, positively evaluated in the alphanumeric-focused works of Nguyen et al. [22] and Zhang et al. [23]. The keypad is mapped directly onto the VR controllers -– thumb-operated trackpads host numbers while the remaining controllers' buttons are used for supporting operations. Thus there is no need for additional user interface (UI) elements in the scene, potentially giving application creators more freedom in designing immersive environments. Moreover, similarly to Tilt-click method [57], this keyboard also does not require the user to look at it when providing input. Therefore, we aim to see how this UI-less technique stands in comparison to others.

Circles: This keyboard (Figure 1 (i)) draws inspiration from the rotary dial known from old telephones, and tilting-controlled keyboards by Čejka et al. [61]. The user is presented with two dials-—the left one for numbers, and the right one for supporting operations—that are controlled by the controller rotations. Thus, the user rotates the controllers along their longitudinal axis, and this rotational movement is translated into the equivalent movement of circular dials. To write numbers, the user rotates them into the highlighted part of the dial and then presses the controller's trigger button. The advantage of this method is the reason we included it—it relies solely on rotations, i.e., the user does not have to aim nor precisely position the controller.

Dice: As the name may reveal, this keyboard, inspired by Keycube [14, 15], employs gamification concepts aiming to entertain the user while performing the desired task (Figure 1 (j)). In this case, the left controller is replaced with a 10-sided dice which movements correspond to the movements of the controller. The numbers are then written by selecting a dice number with the right controller's ray. Moreover, this method also employs simple swinging gestures to perform supporting operations. More precisely, the horizontal swing of the left controller is used to write a minus sign, while stabbing-like movement results in a decimal point being written. The right controller horizontal and vertical swings are used to move the caret and remove the last written character, respectively. The motivation behind Dice is to see the results of such an unusual approach that might be well suitable for game-like scenarios, or even as a general selection technique where the numbers would be replaced with menu items, for example.

3.3. Discarded Keyboards

Apart from the keyboards described in the previous text, we also implemented two additional keyboards, pictured in Figure 2, that were discarded after the pilot study (described in Supplementary material) due to their poor performance results.

Handwriting keyboard: This keyboard uses a ray emitted from the controller to write text like with a classic writing instrument. This input method was explored by other researchers as well, showing some potentially interesting benefits. However, the performance of this method was far from satisfactory in our scenario.

Telephone-like keyboard: This keyboard is based on the keypad of classic mobile phones. The keyboard consists of 11

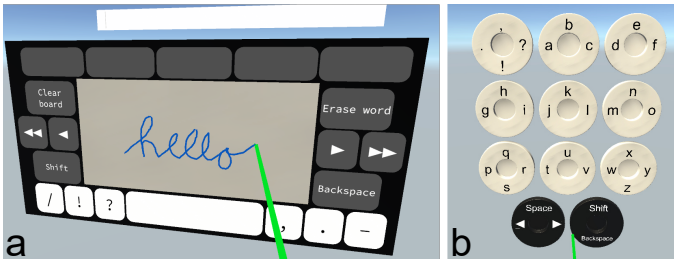

Fig. 2. Two keyboards discarded after the pilot study: a) handwriting keyboard b) telephone-inspired keyboard.

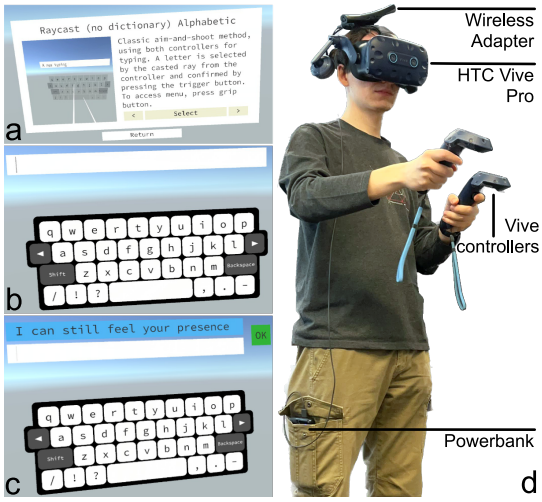

Fig. 3. (left) Screenshots from the application environment showing the (a) keyboard introduction screen, (b) playground scene, and (c) the layout of the testing mode showing a sentence to write. (right) User study setup from the equipment perspective.

circular buttons in total, each of which offers up to four actions. The act of writing is realized by aiming the ray at the desired button, and then by pressing a corresponding part of the trackpad to determine the letter to write. Unfortunately, this method turned out to be too slow and confusing for the users, and thus we did not further proceed with it.

4. Comparison Study Design

All keyboards were first evaluated in a pilot study that was later followed by a final comparison study evaluating improved keyboard designs. We have decided to split the evaluation into two groups, one focusing on alphabetic keyboards, and the other one on numeric ones. This enabled us to keep the same study design but significantly shorten its length for the participants, resulting in less fatigue and a more pleasant experience.

4.1. Participants

All participants were required to be more than 18 years old and were informed that the collected data were anonymous and that there was no reward for participation. Overall, the study involved 40 participants (N = 40, age ranging from 20 to 47 with a mean of 26.13 and standard deviation 6.25, female/male = 13/27), out of these 20 (N = 20, mean age 26.9, female/male = 7/13) tested alphabetic keyboards and 20 (N = 20, mean age 25.35, female/male = 6/14) numeric ones. All of them evaluated their English level as B1 or better (based on the Common European Framework of Reference), but none were native English speakers. Eight users had no prior VR experience, 22 users use VR at least once a year, nine on a monthly basis, and one participant uses it at least once a week. All participants successfully completed the entire testing sessions, which typically lasted around an hour.

4.2. Testing Environment

The evaluated methods were implemented in an application based on the Unity engine, targeting the HTC Vive headsets and corresponding controllers (Figure 3 (right)). The testing environment within this application consisted of three main parts, as shown in Figure 3 (left). First, before testing a specific method, the users were provided with a textual overview of its functionality, serving as an initial introduction to the method. Then, they entered a ”playground” mode, containing a single input field, enabling the users to get familiar with the keyboard without any distractions. Finally, the testing mode itself consisted of two input fields-—the upper one presenting the text to write, and the lower one user’s input. The scene also contained a button used to confirm the current input, and subsequently proceed to the next sentence or number. The input fields were uniformly sized and positioned and designed to fit the whole target text without overflowing the input area. Further, they were horizontally aligned to simplify the verification of text correctness. If an overflow occurred—for example, due to the user’s error—the overflowing text was hidden, and arrows were shown on the left or right part of the input field, suggesting that there is some hidden text the user should know about.

4.3. Procedure

The testing procedure consisted of three main phases–-briefing, study, and debriefing. While each participant went through briefing and debriefing once, the study phase was repeated for every tested keyboard. The order of keyboards was different for each participant-—to reduce the order-effects and carry-over effect-—and was generated using the Balanced Latin Square methodology [62, 63]. The study itself took place in our laboratory with only the supervisor and participant present at the given time. The procedure and questionnaires were equal for alphabetic and numeric groups, with a slight difference in the debriefing form, where the users provided feedback in relation to the type of keyboards they tried.

During the first phase, each participant was introduced to the study goals and tasks that would be performed. Then, the participant signed an informed consent form, followed by filling in a demographic survey form to collect demographic data. Finally, we explained to the participant the basics of the hardware that will be used throughout the study.

The study phase consisted of three main parts. As the first step, the participant was informed about the core idea behind the keyboard to be tested. Then, a ”playground” mode followed. The goal of this mode was to make sure that the user achieves basic familiarity with the method — they went through

all important keys to be able to properly create, remove, and correct text, as well as tried to write a simple text or number. When this mode was over, thus the participant confirmed that they understood how to use the keyboard, the testing mode was initiated. During the testing, the user was shown a sentence or a number, tried to rewrite it, and then confirmed the output to continue with the next phrase. This was repeated five times. The sentences, respectively numbers, were predefined, and stayed the same for all tested methods. In the case of alphabetic input, the sentences were extracted from the MacKenzie's phrase set [64], while the numbers for numeric input keyboards were manually designed. The user's performance for individual phases was measured. When the participant finished with the last input, they exited the virtual reality experience and continued by filling in a questionnaire related to the keyboard that was just tested. This form consisted of System Usability Scale [65, 66] questions, as well as custom-designed questions acquiring additional feedback from the keyboard evaluation perspective. When the form was submitted, the user continued with the next keyboard.

After evaluating all five methods, the participant finalized the study by filling in a debriefing form. This form was designed to provide final insights on all of the methods, as well as generally on the topic of virtual reality text entry.

4.4. Metrics

Several metrics were used to evaluate the objective performance of the keyboards. In their definitions below, P stands for the presented text, i.e., the text the user is supposed to enter, and T stands for the final text inputted by the user. $|P|$, respectively $|T|$, corresponds to the length of these texts. Then, S is the number of seconds the user took to transcribe the P.

- Words Per Minute (WPM) describes the writing speed of a particular alphabetic method measured with respect to five-character words. It is computed as $\frac{|T|-1}{S} \times 60 \times \frac{1}{5}$.
- Characters Per Minute (CPM) is used to compare the speed of numeric methods. It is computed as $\frac{|T|-1}{S} \times 60$.
- Levenshtein distance (LD) is the minimum number of single-character operations needed to transform T into P, i.e., a metric describing how different two strings are. We also employ so-called Levenshtein distance Space Trimmed (LDST) metric, being equal to LD of T and P with whitespace characters removed.
- Corrected (CER) and uncorrected (UER) error rates take into consideration errors occurring during the input of phrase T and fixed before submitting the result (CER), respectively ignored and submitted erroneously (UER). Both of these metrics are included in the outcome of the total error rate (TER) metric.

5. Comparison Study Results

In order to articulate our findings, we have divided our summary of results into two sections by type of feedback—performance (Subsection 5.1) and subjective ratings (Subsection 5.2). We also divide these further based on keyboard types: alphabetic and numeric. Finally, we compare the performance and subjective ratings in the subsequent discussion (Section 6).

5.1. Performance

In order to examine the differences in WPM and various features of the error rates, we conducted a series of parametric tests including Repeated Measures Analysis of Variance (ANOVA), paired samples t-test, and correlations. Alpha was set to 0.05, and all ANOVAs and t-tests were two-sided. Means with 95% confidence intervals are reported in Figure 4 for alphabetic keyboards and Figure 5 for numeric keyboards (standard deviations are available as tables in Supplementary Material).

We decided to examine the error rate via multiple tests in order to fully understand how users were interacting with the keyboards. In this aspect, TER/CER are influenced by errors that may have been fixed before the phrase has been submitted, while LD(ST) focuses only on the final result itself. Selected results (WPM, CPM, TER, LD, LDST) are reported below, while additional details might be found in Supplementary Material.

5.1.1. Alphabetic Keyboards

Overall findings indicate that the different alphabetic keyboards induce different performances in participants, and while one keyboard may dominate in a particular metric, it may perform worse in a different one.

Words Per Minute (WPM): To check for statistically significant differences in WPM, a repeated measures ANOVA was conducted comparing the five different keyboard types. Participants did have significantly different WPM, $F(4, 76) = 166.861, p < 0.001, \eta^2 = 0.898$.

In order to see more details on how the WPM was different across different keyboard types, a series of ten paired samples t-test were conducted. These ten tests compared each keyboard type to every other keyboard type. The only pair that did not show a statistically significant difference in WPM was Swipe and Zoom after Bonferroni correction, $t(19) = 0.458, p = 0.757$ (as a reminder: after Bonferroni correction, the alpha level is 0.005 as the usual 0.05 is divided by the number of paired analyses- 10 in this case). This seems logical in conjunction with data in Figure 4, where we see that the mean WPM values of Swipe and Zoom are separated by less than one WPM. Overall, Speech to Text yielded the highest average WPM (36.027), at almost double the mean score of the second place keyboard, Raycast alphabetic (18.682).

Total Error Rate (TER): Similarly to WPM, a repeated measures ANOVA was conducted comparing the TER of different keyboard types, confirming that participants did have significantly different TER, $F(4, 76) = 45.907, p < 0.001, \eta^2 = 0.707$.

As for the conducted paired samples t-test, the only two pairs that did not show a statistically significant differences after Bonferroni correction in TER was Raycast alphabetic and Speech to Text, $t(19) = 0.599, p = 0.556$ and Speech to Text and Zoom $t(19) = 2.843, p = 0.010$. Once again, this seems logical in conjunction with data in Figure 4, where we see that the mean TERs of Raycast alphabetic and Speech to Text are within a range of one. Overall, Zoom yielded the lowest mean

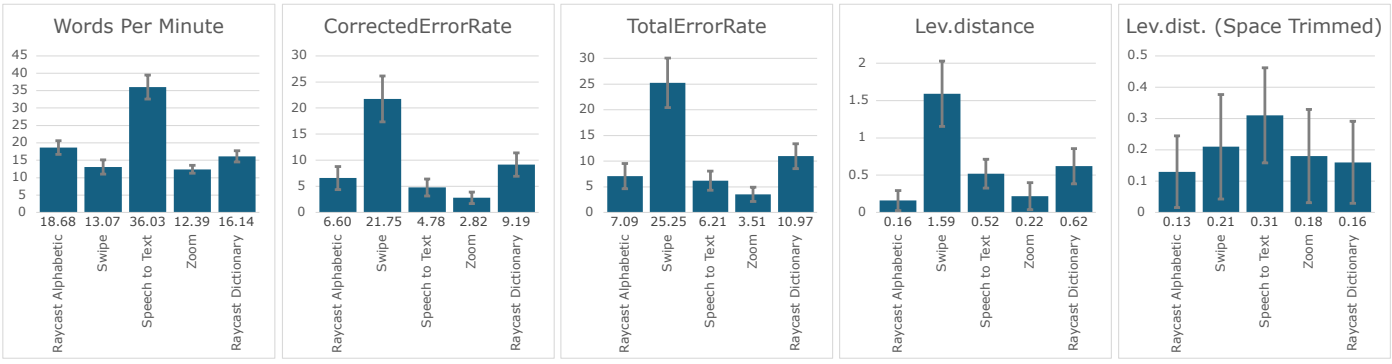

Fig. 4. The performance results for the alphabetic keyboard types. The bars indicate the mean performance for particular measures listed in the chart captions. Whiskers depict the 95% confidence intervals for the means.

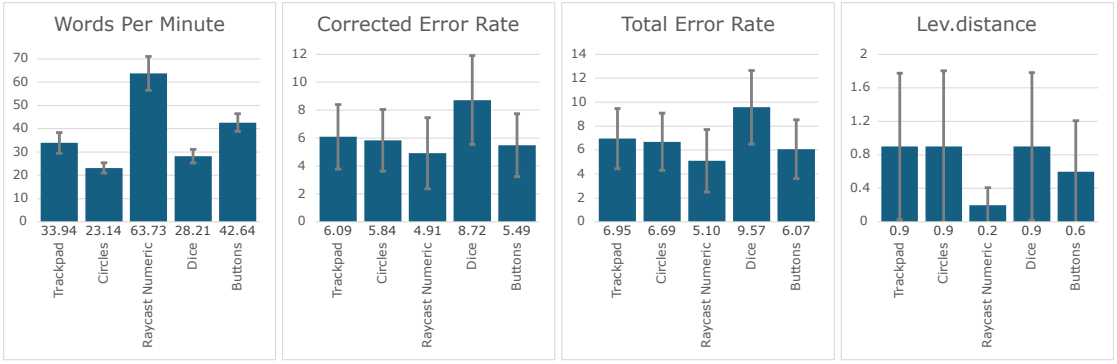

Fig. 5. The performance results for the numerical keyboard types. The bars indicate the mean performance for particular measures listed in the chart captions. Whiskers depict the 95% confidence intervals for the means.

TER (3.511), while Swipe had the highest mean total error rate (25.250).

Levenshtein distance (LD) and Lev. dist. Space Trimmed (LDST): We compared LD and LDST to see if there are any keyboards that seem to noticeably improve in the Space Trimmed version. We ran five paired samples t-test, one for each type of keyboard, and also linear correlations of each pair. Three pairs had statistically significant differences in the t-test: Swipe $t(19) = 7.290, p < 0.001$, Speech to Text $t(19) = 4.368, p < 0.001$, and Raycast dictionary $t(19) = 4.886, p < 0.001$. The correlations of these three pairs were also less than Raycast alphabetic ($r = 0.945$) and Zoom ($r = 0.976$); with Swipe's correlation being $r = 0.436$, Speech to Text's being $r = 0.849$, and Raycast dictionary's being $r = 0.545$.

All these three keyboards provided "automatic space-addition" feature in some way. For example, Swipe keyboard appended a new space every time a word-drawing gesture was finished, and thus when a new word was generated. Based on these results and observations from the study itself, we believe that this type of feature might have negatively influenced error rates, as well as regular LD outcomes. Thus, a more advanced implementation of the space-addition might have improved some aspects of the aforementioned keyboards.

5.1.2. *Numeric Keyboards*

For the evaluation of numeric keyboards, we omitted the LDST metric as there were no whitespaces present in the final number. Overall findings indicate that the different numeric keyboards somewhat induce different performances in participants. There is a clear difference in CPM, though the error rates show a more complicated picture.

Characters Per Minute (CPM): A repeated measures ANOVA was conducted comparing CPM of the five different keyboard types, with the results confirming significantly different CPM, $F(4, 76) = 116.113, p < 0.001, \eta^2 = 0.859$. Similarly, a series of ten paired samples t-test were conducted, where all pairs resulted in statistically significant differences. Overall, Raycast Numeric yielded the highest average CPM, while the lowest average CPM keyboard was Circles (as seen in Figure 5).

Total Error Rate (TER): Based on the results of the repeated measures ANOVA, we may conclude that participants did not show significantly different TER, $F(4, 76) = 2.224, p = 0.074, \eta^2 = 0.105$. A series of ten paired samples t-test followed, revealing that no pairs showed a statistically significant difference in TER. As we can see in Figure 5, Dice had the highest mean TER, while Raycast and Buttons had the lowest one.

5.2. *Subjective Results*

To examine the differences in how participants rated the keyboard types on ten Likert scale questions, we conducted a series of nonparametric tests, again including Friedman test and post hoc Paired-sample Wilcoxon tests. Alpha was set to 0.05 accross all tests. Means with 95% confidence intervals are re-

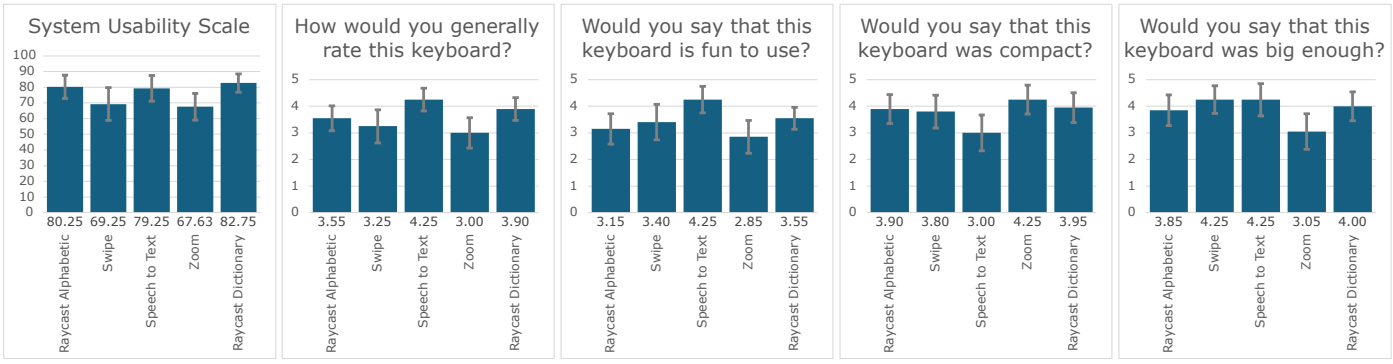

Fig. 6. The subjective ratings for alphabetic keyboard types. The bars indicate the mean value for particular ratings listed in the chart captions. Whiskers depict the 95% confidence intervals for the means.

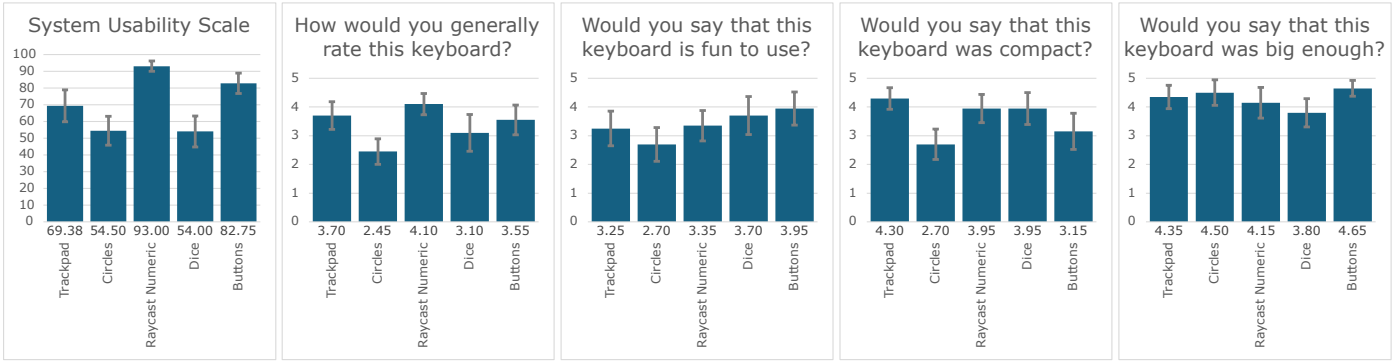

Fig. 7. The subjective ratings for numeric keyboard types. The bars indicate the mean value for particular ratings listed in the chart captions. Whiskers depict the 95% confidence intervals for the means.

ported in Figure 6 for alphabetic keyboards and Figure 7 for numeric keyboards (standard deviations are available as tables in Supplementary Material).

These subjective questions took the form of the System Usability Scale (SUS), which is a ten-item Likert-style questionnaire [65, 66] and four additional Likert-style questions. The SUS is designed to range from 0 to 100, while the other Likert questions have scores that will range from one to five.

5.2.1. Alphabetic

Hereafter we present Friedman tests results, which indicated statistically significant differences. Interestingly, the outcomes of most of the conducted post-hoc Paired-sample Wilcoxon's tests indicated no statistical significance between pairs after Bonferroni correction. This is unsurprising, since with ten pairs the alpha level cut-off shifted from 0.05 to 0.005. These pair analyses can be found in the Supplementary material. In the case of the Friedmans tests of the following measures, participants did rate the keyboards differently.

SUS Scores ($\chi^2(4, n = 20) = 10.367, p = 0.035$): Raycast dictionary received the highest average rating, while the lowest average-rated keyboard was Zoom.

"How would you generally rate this keyboard?" ($\chi^2(4, n = 20) = 14.525, p = 0.006$): Speech to Text received the highest average rating, while the lowest average rated keyboard was Zoom.

"Would you say that this keyboard is fun to use?" ($\chi^2(4, n = 20) = 12.023, p = 0.017$): Again, Speech to Text dominated the average ratings, while the Zoom remained lowest rated.

"Would you say that this keyboard was compact?" ($\chi^2(4, n = 20) = 11.566, p = 0.021$): This time, the tide turned and Zoom achieved the highest average rating, while the lowest average rated keyboard was Speech to Text. The results of Zoom do not come as a surprise since this keyboard was designed with compactness in mind. The lowest rating for Speech to Text may lie in the fact that, in practice, this keyboard requires no more than a single button, potentially not even visible in the UI. However, in our case, the user was given some additional controls, as well as word suggestions, thus the keyboard occupied more space than seemed necessary.

"Would you say that this keyboard was big enough?" ($\chi^2(4, n = 20) = 13.699, p = 0.008$): Overall, Speech to Text and Swipe tied for the highest average rating, while the lowest average-rated keyboard was Zoom. In the post-hoc Paired-sample Wilcoxon's tests, the pairing of Swipe and Zoom was the only test which indicated statistically significant differences after Bonferroni correction.

5.2.2. Numeric

Similarly to alphabetic keyboards, this section presents outcomes of Friedman tests targeting subjective results of numeric methods. All of the Friedman tests indicated statistical significance. We also conducted post-hoc Paired-sample Wilcoxon's

tests, which showed some statistically significant differences even after Bonferroni correction. These tests can be found in the Supplementary material. For each of the Friedmans tests of the following measures, the participants rated the keyboards differently.

SUS Scores ($\chi^2(4, n = 20) = 49.472, p < 0.001$): Raycast numeric had the highest average rating, while the lowest average rated keyboard was Dice.

"How would you generally rate this keyboard?" ($\chi^2(4, n = 20) = 22.710, p < 0.001$): Raycast numeric had the highest average rating, while the lowest average rated keyboard was Circles.

"Would you say that this keyboard is fun to use?" ($\chi^2(4, n = 20) = 17.377, p = 0.002$): Buttons had the highest average rating, while the lowest average rated keyboard was Circles. While we assumed the Buttons keyboard to be fun, we expected that Dice might dominate this question, yet it ended up second.

"Would you say that this keyboard was compact?" ($\chi^2(4, n = 20) = 26.993, p < 0.001$): Trackpad Number had the highest average rating—as expected, since this keyboard was designed to be compact—while the lowest average rated keyboard was Circles.

"Would you say that this keyboard was big enough?" ($\chi^2(4, n = 20) = 12.345, p = 0.015$): Buttons had the highest average rating, while the lowest average rated keyboard was Dice.

6. Discussion

Overall, we found that different keyboards in VR led to different outcomes, both in performance and subjective ratings. In this chapter, we will discuss the most interesting outcomes, especially in relation to our research questions:

- **RQ1**: What are the best and worst performing methods in terms of writing speed and error rate, and what aspects mostly define their performance?
- **RQ2**: How strongly users' subjective experience (i.e., personal keyboard preference) corresponds to the objective data (i.e., measured keyboard performance)?

6.1. The Good, the Bad and the Ugly

This section discusses the RQ1, mostly based on the performance results (Subsection 5.1), and provides possible explanations and recommendations based on the outcomes we discovered.

6.1.1. Alphabetic Keyboards

Among the alphabetic keyboards, Speech to Text was clearly dominant in terms of writing speed (i.e., WPM). Participants also positively accepted it, noting that the speech recognition worked really well. Given the participant pool consisting of non-native English speakers, we were positively surprised by the great results of the Microsoft Azure-backed recognition. As mentioned by some participants, the disadvantage of Speech to Text is that it may feel strange for some to talk to computer or they might feel nervous if other people are present. To counter this disadvantage, it might make sense to include Speech to Text as an addition to traditional VR keyboards. In comparison to existing data, the performance of our Speech to Text keyboard (36.03 WPM) is noticeably better than the one reported in the evaluation of comparable speech-to-text method SWIFTER [45], achieving 23.6 WPM on average. This could be due to many factors, such as advancements in speech-to-text technology, better text suggestions, or different study design. The results of Raycast dictionary and Swipe might be seen as slightly disappointing. While both methods should advance the basic Raycast alphabetic, they were objectively worse in our scenario. In the case of Raycast dictionary, the participants tended to lose precious time by searching for correct suggestion, and if it was not present, they had to resort to continuing to write the given word, whereas with regular Raycast alphabetic, they focused entirely on writing. In any case, the results of the dictionary-backed method are largely influenced by chosen sentences and dictionary. Implementing some more advanced, context-aware techniques, would have a high potential to noticeably improve the result. In any case, the results of Raycast alphabetic (18.68 WPM) generally correspond to the number (16.65 WPM) reported by Boletsis et al. [18], suggesting a rather predictable performance of this method. This also aligns with the mean 15.44 WPM reported by Speicher et al. [21].

In the case of Swipe, we observed an interesting phenomenon. There were few participants who had experience with this type of input from their smartphone, and they managed to use this keyboard very naturally and with high efficiency. On the other hand, some participants had serious troubles with gesture-drawing, as they rarely managed to convince the keyboard to output the word they expected. So, experience seems to play a large role in Swipe performance. This aligns with the observations of Chen et al. [47]. Their Swipe-style keyboard generally achieved slightly better performance than ours—16.43 WPM in comparison to 13.07 WPM. However, when they evaluated the keyboard with the expert user, they measured a significantly better performance of 35.44 WPM. A similar experience situation may apply to Zoom keyboard—since the users see only part of the keyboard layout, having a natural understanding of key placement strongly influences the writing ability.

6.1.2. Numeric Keyboards

In the area of numeric keyboards, Raycast numeric was certainly the fastest method to use, presenting a well-known layout and natural controls. Buttons method also scored well, being reasonably fast and fun at the same time. In this case, implementing a more standardized layout, as well as adding height-adjustability, would potentially further improve the writing speed. Trackpad also scored acceptably well, often being praised for providing tangible feedback not requiring to look at the keyboard. What came as a surprise for us was the performance of Dice when compared to Circles. We expected the Dice to be the slowest, albeit potentially fun, method, and thus the fact that it turned out to be faster than Circles was surprising.

However, the reason for this might lie in the implementation—while Circles keyboard was comfortably usable for the authors, many of the participants had issues when trying to select a number, and so they had to click multiple times to confirm the selection. This negatively influenced the speed of the keyboard, and as some participants noted: the idea might work well with better execution. Thus, we believe the method may achieve noticeably better results if properly adjusted for regular users.

6.2. Performance Versus Subjective Ratings

In this section, we discuss RQ2, i.e., agreements and disagreements between participant's performance and subjective ratings. The statistical significance determinations were also largely impacted by Bonferroni corrections to the alpha level, so many pairs which appeared to be statistically significant initially were found to not actually be significantly different under closer scrutiny. However, the pairs which did still differ substantially provide us with an interesting perspective on keyboard perception.

6.2.1. Alphabetic Keyboards

In terms of performance, Speech to Text produced the highest WPM scores, while Zoom and Swipe were the lowest ones. However, Zoom had the lowest error rate, while Swipe had the highest. These results would imply that Speech to Text and perhaps Zoom would be highly rated by participants, assuming that participant attitude is informed by performance. In some cases, this is consistent, with participants giving Speech to Text the highest average ratings for the General rating and "This keyboard was fun to use" rating. So there appears to be agreement between performance and subjective ratings here.

However, Zoom was rated as having the lowest average rating on the System Usability scale, indicating that it was less usable than the other keyboards. Zoom also had the lowest general rating and "fun" rating. This outcome is rather interesting since Zoom leads to less error, but is not as well liked as the other, more error-prone keyboards. Interesting observations can also be revealed when comparing Raycast alphabetic and Raycast dictionary methods. From the performance perspective, the former resulted in better results in both writing speed and error rates. However, subjective results speak in favor of Raycast dictionary. So, albeit being slower and more error-prone, the users felt the dictionary-enhanced method was actually better. This might be due to the non-intrusiveness of the dictionary feature—the participants selected suggested words at their will, so their writing experience was not hindered in any way. Moreover, since all of the participants were non-native speakers, the fact that the dictionary suggested correctly spelled words might have been an additional appreciated factor. All in all, there is some agreement between the performance and subjective ratings for the alphabetic keyboards but we may also find interesting differences.

6.2.2. Numeric Keyboards

For CPM performance, Raycast numeric produced the highest score, while Circles produced the lowest one. Raycast also yielded the lowest error rates, while Dice had the highest one. There seems to be some agreement in the subjective ratings, with Raycast having the highest ratings on the SUS scores and having the highest average general rating. Similarly, Circles had the lowest WPM and was the least fun to use. Dice had higher error rates than Circles and was also rated as less usable in the SUS. So, for the numeric keyboards, there is solid agreement between the performance and subjective ratings.

6.3. Limitations and Future Work

This section discusses the limitations of our study, starting with implementations of the methods. As already briefly discussed, some of the presented techniques might be further improved to deliver better results. Namely, Raycast dictionary and Swipe methods would benefit from a more advanced suggestion algorithm, taking into consideration contextual information. Similarly, the Buttons technique for numeric input could be improved—the chosen layout of numbers seemed confusing to some users, so a more common ITU-T E.161 standard layout might be a better choice. Similarly, this keyboard can be enhanced with the ability to adjust the height at which it is positioned, as in our scenario, the fixed height was challenging for some study participants. Layout improvements might be worth exploring also in the case of Dice keyboard, where a different placement of numbers may impact performance. Finally, Circles method could benefit from a slightly more polished implementation where the selection of highlighted numbers is more tolerant with regard to the user's precision.

Additional limitations are pertaining to the study design. First, as already mentioned, our study focused on alphabetic and numeric inputs separately, and thus we do not provide conclusive results for alphanumeric typing. In this direction, further research might be needed, possibly using our outcomes as a basis for the design of the combined methods. Then, it is good to note that our study measured the performance of methods in a scenario where users were exposed to the individual methods for a rather short time. Thus, we could not evaluate the long-term performance of these methods, i.e., the results of methods with a steeper learning curve, like Swipe, might turn out to be more promising as the user's experience increases. Similarly, keyboards can be evaluated also from the perspective of single-use long-term performance, i.e., how suitable they are for longer writing sessions with respect to fatigue and related factors. Finally, our study focused exclusively on textual inputs using the Latin alphabet, and as such, we cannot predict the suitability of individual techniques for different alphabets.

Heading to the future, it would be interesting to compare the performance of numeric keyboards with the numeric keypad layout, known from computers, and telephone keypad layouts. Then, we expect that promising research discoveries might be achieved when exploring the long-term performance of each individual keyboard type. Similarly, exploring the different ways of visual or haptic feedback during the keyboard interaction might lead to valuable insights into the methods' performance and usability. In the case of Speech to Text keyboard, it might be intriguing to study the influence of the virtual environment on the users' comfort when speaking. Finally, a more in-depth look at the Swipe keyboard performance with respect to the user's experience might reveal valuable insights.

7. Conclusion

Textual input is an important aspect of interaction with modern electronic devices. It enables to control the digital experiences, as well as to obtain, create, and exchange knowledge with others. The importance of alphabetic and numeric input applies also in the context of head-mounted virtual reality, being a medium providing dedicated means of interaction, having both benefits and limitations when compared to traditional computer hardware.

Therefore, in this study, we implemented and subsequently evaluated five alphabetic and five numeric input methods for head-mounted virtual reality. The methods either replicated established keyboard designs, drew inspiration from techniques found in other hardware devices, or were designed by us with a specific goal in mind. In all cases, the presented keyboards were designed to use only standard hardware provided with the VR headsets. By evaluating the methods in a two-group study with 40 participants, we were able to collect both objective and subjective performance data. This enabled us to more deeply understand the efficiency of the methods, observe if we can replicate results from other researchers, as well as to discuss the possible reasoning behind the outcomes.

References

[1] Fourrier, N, Moreau, G, Benaouicha, M, Normand, JM. Handwriting for efficient text entry in industrial vr applications: Influence of board orientation and sensory feedback on performance. IEEE Transactions on Visualization and Computer Graphics 2023;29(11):4438–4448. doi:10.1109/TVCG.2023.3320215.

[2] Motejlek, J, Alpay, E. Taxonomy of virtual and augmented reality applications in education. IEEE Transactions on Learning Technologies 2021;14(3):415–429. doi:10.1109/TLT.2021.3092964.

[3] Aleven, V, Ogan, A, Popescu, O, Torrey, C, Koedinger, K. Evaluating the effectiveness of a tutorial dialogue system for self-explanation. In: Intelligent Tutoring Systems: 7th International Conference, ITS 2004, Maceió, Alagoas, Brazil, August 30-September 3, 2004. Proceedings 7. Springer; 2004, p. 443–454.

[4] Lee, Y, Kim, GJ. Vitty: Virtual Touch Typing Interface with Added Finger Buttons. In: Lackey, S, Chen, J, editors. Virtual, Augmented and Mixed Reality; vol. 10280. 2017, p. 111–119. doi:10.1007/978-3-319-57987-0_9.

[5] Menzner, T, Otte, A, Gesslein, T, Grubert, J, Gagel, P, Schneider, D. A Capacitive-sensing Physical Keyboard for VR Text Entry. In: 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). 2019, p. 1080–1081. doi:10.1109/VR.2019.8797754.

[6] Otte, A, Menzner, T, Gesslein, T, Gagel, P, Schneider, D, Grubert, J. Towards Utilizing Touch-sensitive Physical Keyboards for Text Entry in Virtual Reality. In: 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). 2019, p. 1729–1732. doi:10.1109/VR.2019.8797740.

[7] Gugenheimer, J, Dobbelstein, D, Winkler, C, Haas, G, Rukzio, E. 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. UIST '16; 2016, p. 49–60. doi:10.1145/2984511.2984576.

[8] Lee, J, Kim, B, Suh, B, Koh, E. Exploring the Front Touch Interface for Virtual Reality Headsets. In: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems. 2016, p. 2585–2591. doi:10.1145/2851581.2892344.

[9] Hutama, W, Harashima, H, Ishikawa, H, Manabe, H. HMK: Head-Mounted-Keyboard for Text Input in Virtual or Augmented Reality. In: The Adjunct Publication of the 34th Annual ACM Symposium on User Interface Software and Technology. 2021, p. 115–117. doi:10.1145/3474349.3480195.

[10] Son, J, Ahn, S, Kim, S, Lee, G. Improving Two-Thumb Touchpad Typing in Virtual Reality. In: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems. 2019, p. 1–6. doi:10.1145/3290607.3312926.

[11] Witt, H, Janssen, T. Comparing two methods for gesture based short text input using chording. In: CHI '07 Extended Abstracts on Human Factors in Computing Systems. 2007, p. 2759–2764. doi:10.1145/1240866.1241075.

[12] Gupta, A, Ji, C, Yeo, HS, Quigley, A, Vogel, D. RotoSwype: Word-Gesture Typing using a Ring. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 2019, p. 1–12. doi:10.1145/3290605.3300244.

[13] Meier, M, Streli, P, Fender, AR, Holz, C. Demonstrating TapID for Rapid Touch Interaction on Surfaces in Virtual Reality for Productivity Scenarios. In: Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems. 2021, p. 1–4. doi:10.1145/3411763.3451553.

[14] Brun, D, Gouin-Vallerand, C, George, S. Keycube is a Kind of Keyboard (k^3). In: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems. 2019, p. 1–4. doi:10.1145/3290607.3313258.

[15] Brun, D, George, S, Gouin-Vallerand, C. Keycube: Text Entry Evaluation with a Cubic Device. In: Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems. 2020, p. 1–9. doi:10.1145/3334480.3382837.

[16] Daydream Labs: exploring and sharing VR's possibilities. 2016. URL: https://blog.google/products/google-ar-vr/daydream-labs-exploring-and-sharing-vrs/.

[17] Boletsis, C, Kongsvik, S. Text Input in Virtual Reality: A Preliminary Evaluation of the Drum-Like VR Keyboard. Technologies 2019;7(2):31. doi:10.3390/technologies7020031.

[18] Boletsis, C, Kongsvik, S. Controller-based Text-input Techniques for Virtual Reality: An Empirical Comparison. International Journal of Virtual Reality 2019;19(3):2–15. doi:10.20870/IJVR.2019.19.3.2917.

[19] Yanagihara, N, Shizuki, B. Cubic Keyboard for Virtual Reality. In: Proceedings of the 2018 ACM Symposium on Spatial User Interaction. SUI '18; 2018, p. 170. doi:10.1145/3267782.3274687.

[20] Yanagihara, N, Shizuki, B, Takahashi, S. Text Entry Method for Immersive Virtual Environments Using Curved Keyboard. In: Proceedings of the 25th ACM Symposium on Virtual Reality Software and Technology. VRST '19; 2019, p. 1–2. doi:10.1145/3359996.3365026.

[21] Speicher, M, Feit, AM, Ziegler, P, Krüger, A. Selection-based Text Entry in Virtual Reality. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. CHI '18; 2018, p. 1–13. doi:10.1145/3173574.3174221.

[22] Nguyen, A, Bittman, S, Zank, M. Text Input Methods in Virtual Reality using Radial Layouts. In: Proceedings of the 26th ACM Symposium on Virtual Reality Software and Technology. VRST '20; 2020, p. 1–3. doi:10.1145/3385956.3422114.

[23] Zhang, Z, Sun, M, Gao, B, Wang, L. 2-Thumbs Typing: A Novel Bimanual Text Entry Method in Virtual Reality Environments. In: 2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW). 2021, p. 530–531.

[24] Jiang, H, Weng, D. HiPad: Text entry for Head-Mounted Displays Using Circular Touchpad. In: 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). 2020, p. 692–703. ISSN: 2642-5254.

[25] Tominaga, K, Fujita, S, Takakura, R, Shizuki, B. Investigating the Effects of Position and Angle of Virtual Keyboard on Text Entry Performance and Workload. In: Asian CHI Symposium 2021. Asian CHI Symposium 2021; 2021, p. 25–27. doi:10.1145/3429360.3468174.

[26] Dube, TJ, Arif, AS. Impact of Key Shape and Dimension on Text Entry in Virtual Reality. In: Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems. CHI EA '20; 2020, p. 1–10. doi:10.1145/3334480.3382882.

[27] Yang, Z, Chen, C, Lin, Y, Wang, D, Li, H, Xu, W. Effect of spatial enhancement technology on input through the keyboard in virtual reality environment. Applied Ergonomics 2019;78:164–175. doi:10.1016/j.apergo.2019.03.006.

[28] Knierim, P, Schwind, V, Feit, AM, Nieuwenhuizen, F, Henze, N. 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. CHI '18; 2018, p. 1–9. doi:10.

1145/3173574.3173919.
[29] Grubert, J, Witzani, L, Ofek, E, Pahud, M, Kranz, M, Kristensson, PO. Effects of Hand Representations for Typing in Virtual Reality. In: 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). 2018, p. 151–158.
[30] Wan, T, Wei, Y, Shi, R, Shen, J, Kristensson, PO, Atkinson, K, et al. Design and evaluation of controller-based raycasting methods for efficient alphanumeric and special character entry in virtual reality. IEEE Transactions on Visualization and Computer Graphics 2024;:1–11doi:10.1109/TVCG.2024.3349428.
[31] Li, Y, Sarcar, S, Zheng, Y, Ren, X. Exploring Text Revision with Backspace and Caret in Virtual Reality. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. CHI '21; 2021, p. 1–12. doi:10.1145/3411764.3445474.
[32] Ogitani, T, Arahori, Y, Shinyama, Y, Gondow, K. 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). 2018, p. 342–349.
[33] Adhikary, J, Vertanen, K. Text Entry in Virtual Environments using Speech and a Midair Keyboard. IEEE Transactions on Visualization and Computer Graphics 2021;27(5):2648–2658.
[34] Tomaru, Y, Tanaka, T. Proposal of Character Input Method in VR Environment using Leap Motion. In: 2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI). 2019, p. 1037–1038.
[35] Jimenez, JG, Schulze, JP. Continuous-Motion Text Input in Virtual Reality. Electronic Imaging 2018;30:1–6. doi:10.2352/ISSN.2470-1173.2018.03.ERVR-450.
[36] Pastor, A. Virtual hands: a comparative study of two text input paradigms for VR. 2020. URL: https://osf.io/fwp8t/.
[37] Nooruddin, N, Dembani, R, Maitlo, N. HGR: Hand-Gesture-Recognition Based Text Input Method for AR/VR Wearable Devices. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2020, p. 744–751.
[38] Komiya, K, Nakajima, T. A Japanese input method using leap motion in virtual reality. In: 2017 Tenth International Conference on Mobile Computing and Ubiquitous Network (ICMU). 2017, p. 1–2.
[39] Fashimpaur, J, Kin, K, Longest, M. 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. CHI EA '20; 2020, p. 1–7. doi:10.1145/3334480.3382888.
[40] Lu, X, Yu, D, Liang, HN, Feng, X, Xu, W. 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). 2019, p. 1060–1061. ISSN: 2642-5254.
[41] Yu, C, Gu, Y, Yang, Z, Yi, X, Luo, H, Shi, Y. 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. CHI '17; 2017, p. 4479–4488. doi:10.1145/3025453.3025964.
[42] Rajanna, V, Hansen, JP. 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. ETRA '18; 2018, p. 1–10. doi:10.1145/3204493.3204541.
[43] Ma, X, Yao, Z, Wang, Y, Pei, W, Chen, H. Combining Brain-Computer Interface and Eye Tracking for High-Speed Text Entry in Virtual Reality. In: 23rd International Conference on Intelligent User Interfaces. IUI '18; 2018, p. 263–267. doi:10.1145/3172944.3172988.
[44] Monteiro, P, Gonçalves, G, Coelho, H, Melo, M, Bessa, M. Hands-free interaction in immersive virtual reality: A systematic review. IEEE Transactions on Visualization and Computer Graphics 2021;27(5):2702–2713. doi:10.1109/TVCG.2021.3067687.
[45] Pick, S, Puika, AS, Kuhlen, TW. SWIFTER: Design and evaluation of a speech-based text input metaphor for immersive virtual environments. In: 2016 IEEE Symposium on 3D User Interfaces (3DUI). 2016, p. 109–112.
[46] Derby, JL, Rickel, EA, Harris, KJ, Lovell, JA, Chaparro, BS. "We Didn't Catch That!" Using Voice Text Input on a Mixed Reality Headset in Noisy Environments. Proceedings of the Human Factors and Ergonomics Society Annual Meeting 2020;64(1):2102–2106. doi:10.1177/1071181320641509.
[47] Chen, S, Wang, J, Guerra, S, Mittal, N, Prakkamakul, S. Exploring Word-gesture Text Entry Techniques in Virtual Reality. In: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems. CHI EA '19; 2019, p. 1–6. doi:10.1145/3290607.3312762.
[48] Kern, F, Niebling, F, Latoschik, ME. Text input for non-stationary xr workspaces: Investigating tap and word-gesture keyboards in virtual and augmented reality. IEEE Transactions on Visualization and Computer Graphics 2023;29(5):2658–2669. doi:10.1109/TVCG.2023.3247098.
[49] Poupyrev, I, Tomokazu, N, Weghorst, S. Virtual Notepad: handwriting in immersive VR. In: Proceedings. IEEE 1998 Virtual Reality Annual International Symposium. 1998, p. 126–132.
[50] Kuś, S, Szmurło, R. CNN-based character recognition for a contextless text input system in immersive VR. In: 2021 22nd International Conference on Computational Problems of Electrical Engineering (CPEE). 2021, p. 1–4.
[51] Elmgren, R. Handwriting in VR as a Text Input Method. Master's thesis; KTH, School of Computer Science and Communication (CSC); 2017.
[52] Takahashi, R, Shirai, S, Orlosky, J, Uranishi, Y, Takemura, H. A Japanese Character Flick-Input Interface for Entering Text in VR. In: 2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct). 2021, p. 251–253.
[53] Swieso, S, Yao, P, Miller, M, Jothi, A, Zhao, A, Zyda, M. Toward Using Machine Learning-Based Motion Gesture for 3D Text Input. In: Proceedings of the 2021 ACM Symposium on Spatial User Interaction. SUI '21; 2021, p. 1–2. doi:10.1145/3485279.3485302.
[54] Li, Z, Peiris, R. RotateEntry: Controller-rolling-style Text Entry for Three Degrees of Freedom Virtual Reality Devices. Frameless 2021;3(1). URL: https://scholarworks.rit.edu/frameless/vol3/iss1/21.
[55] Lepouras, G. Numerical input techniques for immersive virtual environments. In: 2009 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurements Systems. 2009, p. 240–245. ISSN: 1944-9410.
[56] Lepouras, G. Comparing methods for numerical input in immersive virtual environments. Virtual Reality 2018;22(1):63–77. doi:10.1007/s10055-017-0312-5.
[57] Gao, C, Pastel, R, Tan, J. Tilt-click: One-handed eyes-free numeric and symbol input for calculator applications. In: 2012 IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS) Proceedings. 2012, p. 30–33. ISSN: 1944-9410.
[58] de Zoeten, R. Recognizing input for swipe based keyboards. Bachelor thesis; University of Amsterdam; 2013.
[59] Cloud computing services: Microsoft azure. https://azure.microsoft.com; Visited on 04/03/2024.
[60] International Telecommunication Union, . E.161 : Arrangement of digits, letters and symbols on telephones and other devices that can be used for gaining access to a telephone network. https://www.itu.int/rec/T-REC-E.161-200102-I/en; 2001.
[61] Čejka, J, Chmelík, J, Liarokapis, F. Exploring tilting methods for typing under water. Multimedia Tools and Applications 2021;80(20):31085–31103. doi:10.1007/s11042-020-09305-7.
[62] Bradley, JV. Complete counterbalancing of immediate sequential effects in a latin square design. Journal of the American Statistical Association 1958;53(282):525–528. doi:10.2307/2281872.
[63] Masson, D. Balanced latin square online generator. https://cs.uwaterloo.ca/~dmasson/tools/latin_square/; Visited on 04/03/2024.
[64] MacKenzie, IS, Soukoreff, RW. Phrase sets for evaluating text entry techniques. In: CHI '03 Extended Abstracts on Human Factors in Computing Systems. CHI EA '03; 2003, p. 754–755. doi:10.1145/765891.765971.
[65] Brooke, J. Sus: A quick and dirty usability scale. Usability Eval Ind 1995;189.
[66] Brooke, J. SUS: A retrospective. J Usability Stud 2013;8(2):29–40.