Text Entry in Virtual Reality: Implementation of FLIK Method and Text Entry Testbed

Eduardo Soto and Robert J. Teather(✉)

Carleton University, Ottawa, ON, Canada
easm93@gmail.com, rob.teather@carleton.ca

Abstract. We present a testbed for testing text entry techniques in virtual reality, and two experiments employing the testbed. The purpose of the testbed is to provide a flexible and reusable experiment tool for text entry studies, in such a way to include studies from a variety of sources, more specifically to this work, from virtual reality text entry experiments. Our experiments evaluate common text entry techniques and one novel one that we have dubbed the Fluid Interaction Keyboard (FLIK). These experiments not only serve as a way of validating the text entry test-bed, but also contribute the results of these studies to the pool of research related to text entry in virtual reality.

Keywords: Virtual reality · Text entry · FLIK · Words per minute · Head-mounted displays

1 Introduction

According to Ko and Wobbrock, text is a ubiquitous form of verbal communication [8]. Text entry refers to the process of creating messages composed of characters, numbers, and symbols using an interface between a user and a machine. Text entry has been extensively studied in the field of human-computer interaction (HCI) over the past few decades, especially for use in desktop and mobile contexts. Efficient text entry is important because it directly impacts the user's ability to write documents, send messages, and communicate effectively when verbal communication is not available.

Virtual reality (VR) is defined as the use of computer technology to create simulated environments. It often employs head-mounted displays (HMD) along with spatially tracked controllers or hand trackers as input devices. VR has become increasingly popular in recent years; increased consumer demand means the interaction effectiveness of VR systems affects more people than ever. Our work focuses on text entry in VR, such as entering text in search bars or sending text messages. In such use cases, it is important to include an effective option to enter text in VR systems as well as an effective method to evaluate novel text entry techniques and compare with existing techniques.

To type long-form text in a VR environment is not yet feasible or likely even desirable. However, there are many examples where text input of quick messages or short notes is important, even in VR. For example, consider playing a multiplayer online VR game, where communicating in written form may be preferable to speech communication. For

C. Stephanidis et al. (Eds.): HCII 2020, LNCS 12428, pp. 225–244, 2020.
https://doi.org/10.1007/978-3-030-59990-4_18

gaming scenarios, one might find the need to type a quick SMS to a friend inviting them to come online, without interrupting the gameplay. In scientific fields or even in architecture, it might be useful to be immersed in a 3D environment and be able to annotate different components of the environment. In a VR conference call, a user might like to take notes and potentially send back-channel messages to other attendees.

While speech-to-text technologies could work well for some purposes, there are cases where some form of typing in VR might be preferable to voice recognition. Consider, for example, being immersed in VR whilst in a loud environment, or when you need to maintain quiet. For example, VR use has increased in areas such as office work, collaboration, and training, and education. For these applications, inputting text is an essential part of these experiences and more often than not, precise text entry is required rather than using a speech-to-text method, which may disturb other people. Modification of existing text is unreliable and inaccurate as compared to more refined and direct text entry techniques.

The first contribution of this paper is a text entry testbed. A common issue for text entry research is consistency and better standardization of methodological practice in evaluating text entry systems [1]. Methodological consistency is important in text entry research since usually, analysis is comparative, i.e., involves empirically comparing text entry techniques to determine which is most efficient. Employment of similar methods and metrics to ensure comparability between studies is motivating factor of the text entry testbed. The text input testbed is a tool for conducting text entry studies which supports desktop, mobile, VR, and other text input methods, that adheres to standard practice in HCI text input research. The testbed provides a consistent platform to perform text entry comparisons regardless of the techniques used.

The second contribution is a novel VR text entry technique called Fluid Interaction Keyboard (FLIK). FLIK uses two 6-degree of freedom (6DOF) controllers (e.g., Oculus Touch controllers) together with a simple interface for fast and intuitive text entry. It operates by selecting characters from a virtual soft keyboard by directly touching the characters. When eliminating physical button presses on the controllers, users are enabled to type faster using this technique and achieve fluid hand motions to increase the speed of character selection. FLIK was inspired by another text entry technique, Cutie Keys [2, 19] and was designed to improve on existing practice in VR text entry. The third and final contribution is a formal evaluation of FLIK, comparing it to existing VR text entry techniques through two user studies.

2 Related Work

Text entry has been studied for decades. In typical text entry experiments, the time to enter a phrase is recorded while transcribing text to provide a measure of entry speed, while the transcribed text is compared with the original text to measure errors. Entry speed is a key metric for text entry since the goal of text entry techniques is most commonly to offer fast ways of entering text into a system. However, entry speed cannot be looked at by itself; error rate (e.g., number of mistyped characters) is another metric that is used in text entry studies along with entry speed to find a balance between fast text entry speed and acceptable error rates.

Methodology for conducting text entry studies is detailed by Mackenzie [10]; he focuses on the evaluation of text entry techniques and laid out strategies in order to do so. A typical text entry study starts with a working prototype. Users must be able to enter text, and have it displayed as a result. Once this prototype is implemented, substantial pre-experimental testing can begin. The first step is to get a rough idea of the entry speed possible with the technique in question. Next is to decide what the evaluation task will consist of. For text entry studies, the task is commonly to transcribe a set of phrases as quickly and accurately as possibly using some text entry technique as the typing method.

For the methodology described above, a set of phrases is needed to conduct the study. Mackenzie provides such a phrase set for text entry experiments [11] which provides a consistent metric for text entry research independent of technique used, technology involved, or even the main researcher conducting a study. Mackenzie's phrase set consists of 500 phrases to be used in such studies; this provides easy access to much needed source material to be used in text entry.

Wobbrock et al. [20] describe measures of text entry performance in which they include words per minute as a text entry speed measure as well as the minimum string distance as an error rate measure. We now summarize key metrics of text entry performance, several of which we employ in our experiments.

Entry speed, in words per minute, is calculated as seen in Eq. 1, as described by Boletsis et al. [2].

$$wpm = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5} \tag{1}$$

where S is the time in seconds from the first to the last key press and $|T|$ is the number of characters in the transcribed text (i.e., the phrase length). The constant '60' corresponds to the number of seconds in a minute, and the factor of one fifth corresponds to how many characters compose an average word, which is defined to be 5 characters in text entry experiments. This definition makes results more generalizable and consistent across varying phrase sets used for input. We subtract 1 from the length since timing starts as soon as participants enter the first key, hence the first key is not timed, necessitating reducing the phrase length by 1.

An important metric for measuring text entry accuracy is the minimum string distance (MSD) [12, 14–16]. MSD instead is based on an algorithm used in DNA analysis, phylogenetics, spelling correction, and linguistics for measuring the distance between two strings. As an example, consider two strings 'abcd' and 'acbd'. To calculate the error between these two strings you could take the transcribed string and delete the c, then, insert a c after the b. This requires two actions to make the strings identical, so MSD = 2. Minimum String Distance is denoted MSD(A, B) where A is the presented text and B is the transcribed text. With this in mind, MSD(A, B) = 0 means A and B are identical. Further refinement for error analysis [12] notes the difficulty of analyzing errors in text entry systems, in that the displacement of letters in certain text entry errors might make correctly typed letters incorrect.

Text entry techniques are the input methods for entering text into a system. These can include physical keyboards, touchpads, voice recognition, and even pencils. There has been extensive research done in the area of text entry using physical keyboards [5–7]. This work typically focuses on conducting text entry experiments using standard

keyboards to enter text into a computer system, comparing entry speed and accuracy measures. The main focus lies on physical keyboard text entry studies in the context of VR use. One prominent topic in the use of physical keyboards in VR includes hand visualizations when typing on physical keyboards [5–7], and comparing different hand representations in VR text entry. For example, Grubert et al. [6] made use of the Logitech Bridge SDK [18], which was used to track the physical keyboard and hands, and display a digital hand representation in VR. The hand representations included no hands, realistic hands, finger tips only, and VideoHand, where a video pass-through allows the user to see their hands in VR. They found no significant results on entry speed; however, participants preferred VideoHand.

VR text entry techniques usually do not use a physical keyboard, as standard keyboards tend to restrict the user's mobility, while VR systems often require users to physically move around. Thus, the two most common techniques are controller pointing and bimanual text entry. Controller pointing requires spatially tracked controllers to point at and select keys on a soft (i.e., virtual) keyboard, typically via ray-casting. Bimanual text entry presents a soft keyboard split into two sides and requires two touchpads or joysticks to move the cursor on each side of the keyboard to select keys. Each thumb controls one of the two cursors on the corresponding side of the virtual keyboard [13]. Bimanual entry offers entry speeds as high as 15.6 WPM, but on average 8.1 WPM. Similar techniques have been included in several studies, which consistently report low entry speeds between 5.3 and 10 WPM [2, 17].

We finish our discussion of past VR text entry techniques by mentioning BigKey by Faraj et al. [4], as we employ it in our second experiment, and it is a feature supported by our text entry testbed. BigKey resizes keys on the virtual keyboard in real-time. It uses the probability of a letter being typed *next* to increase the size of the predicted next key, and decrease the size of less probable keys. On average, this can increase entry speed, since as described by Fitts' law, larger targets (at otherwise equal distance) offer faster selection times [9].

3 Text Entry in Virtual Reality

This section describes the implementation of our text entry testbed, BigKey, word disambiguation, and the input techniques evaluated in the studies.

3.1 Text Entry Testbed

Noting the need for an experimental framework to conduct text entry studies in VR, we first developed a text entry experiment testbed. The design and implementation of the text entry testbed was targeted as a flexible text entry experiment framework developed with Unity 3D 2018.3.7f1. It was developed for conducting user studies on text entry in VR for the purpose of this work. However, it doubles as a general purpose text entry experiment tool since it can be adapted for use with any (i.e., non-VR) text entry technique. Researchers can develop their text entry techniques and easily integrate it with the text entry testbed for initial performance test, up to full scale user studies.

Starting the testbed, the experimenter is presented with the welcome screen. The experimenter enters a participant ID, the number of phrases that will be presented to the participant in each block, and the number of blocks (i.e., repetitions of all experiment conditions). For example, if 10 phrases are chosen, and 3 blocks are chosen, 30 phrases will be typed overall split into 3 blocks.

Upon proceeding, the experimenter is then prompted to select a text entry method on the next screen. The input technique selection screen presents a list of options which is customized by the experimenter depending on what text entry techniques they include in the evaluation. If the study requires multiple levels of independent variables to choose from, additional options appear allowing the experimenter to choose each condition. Once everything is set and an initial text entry method to test is chosen, the system will start with the first block of the chosen method.

When starting the first block and any subsequent block, a new unique log file is created. In the scene, there are three main components: The phrase to be transcribed, the text entered so far, and the text entry technique are the main components that the participant is able to visualize in the scene. The testbed currently uses Mackenzie's phrase set of 500 phrases as the default phrases, which are commonly used in text entry experiments [11], but these can be changed by replacing a file called 'phrases.txt' with any other phrase set as long as it follows the same format as Mackenzie's phrase set text file. At this point, the main experiment begins, and is seen in Fig. 1.

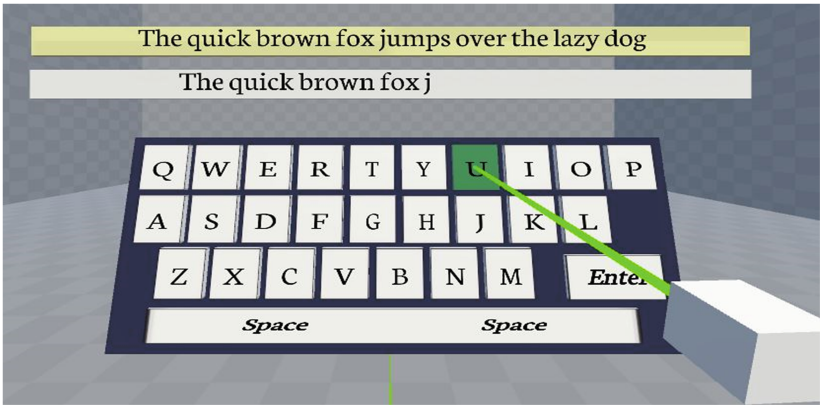

Fig. 1. Elements presented to participants during text entry experiment

As each trial commences, the participant sees the target phrase, the text they have entered so far in a separate text field, and any components that need to be visualized required for a given text entry technique. The testbed randomly selects a phrase from the phrase set; this is the "target" phrase, and it is presented to the participant in the phrase to transcribe section and will wait for the participant to enter the first letter, this is repeated for the number of phrases selected by the experimenter for each block. When the participant enters the first character, a timer starts. The timer stops once the participant hits ENTER or an equivalent key on the virtual keyboard being used. The timer measures how long it took the participant to enter the phrase. While the participant is entering text,

the system records the raw input stream, which corresponds to all characters selected in order; this includes space, backspace, and modifiers.

When the participant completes a phrase and hits ENTER, the testbed records all relevant details to a log file (see Fig. 2). In the log file, each trial appears as a single row, and includes experiment setup details such as the participant ID, text entry technique, current condition, block number, the phrase to be transcribed, the text that is actually transcribed by the participant, the raw input stream, time, the calculated entry speed in words per minute, and error rate in minimum string distance [1].

Participant ID	Text Entry Tecnique	Condition	Block	Phrase	Transcribed	Keystrokes	Time	MSD	Length Phrase	Length Trans	wpm	Error Rate %
1	Ray	None	1	bad for the environment	bad for the environment	bad-for-the-environment	16.1202	0	23	23	16.37696803	0.00%
1	Ray	None	1	we park in driveways	we paek in drieways	we-paek-in-drieways	18.9295	1	20	20	12.04469215	5.00%
1	Ray	None	1	what a monkey sees a monkey will do	what a monkey sees a monkey will do	what-a-monkey-sees-a-monkey-will-do	29.9984	0	35	35	13.60072537	0.00%
1	Ray	None	1	world population is growing	world population is growing	world-population-is-growing	21.6486	0	27	27	14.41201741	0.00%
1	Ray	None	1	what a lovely red jacket	what al lovely red jacket	what-al-lovely-red-jacket	17.1598	1	24	25	16.78341239	4.00%
1	Ray	None	1	an excellent way to communicate	an excellent way to communicate	an-excellent-way-to-communicate	24.3003	0	31	31	14.81463192	0.00%
1	Ray	None	1	he is still on our team	he is still on our team	he-is-still-on-our-team	17.5646	0	23	23	15.03023126	0.00%
1	Ray	None	1	not quite so smart as you think	not quite so smart as you think	not-quite-so-smart-as-you-think	27.0893	0	31	31	13.28937994	0.00%
1	Ray	None	2	my bike has a flat tire	my bike has a flat tire	my-bike-has-a-flat-tire	15.3452	0	23	23	17.20407684	0.00%
1	Ray	None	2	effort is what it will take	effort is what it will take	effort-is-what-it-will-take	22.0915	0	27	27	14.1230701	0.00%
1	Ray	None	2	she wears too much makeup	she wears too much makeup	she-wears-too-much-makeup	18.5272	0	25	25	15.54471264	0.00%
1	Ray	None	2	this equation is too complicated	this equation is too complicated	this-equation-is-too-complicated	20.7602	0	32	32	17.91890252	0.00%
1	Ray	None	2	i took the rover from the shop	i took the rover from the shop	i-took-the-rover-from-the-shop	20.9074	0	30	30	16.6482432	0.00%
1	Ray	None	2	join us on the patio	join us on the patio	join-us-on-the-patio	19.6099	0	20	20	11.62678035	0.00%
1	Ray	None	2	an inefficient way to heat a house	an irefficient way totheat a house	an-irefficient-way-totheat-a-house	28.6977	2	34	34	13.79901525	5.88%
1	Ray	None	2	meet tomorrow in the lavatory	meet tomorrow in the lavatory	meet-tomorrow-in-the-lavatory	27.8705	0	29	29	12.05575788	0.00%
1	Ray	None	3	companies announce a merger	companies announce a merger	companies-announce-a-merger	22.102	0	27	27	14.11636956	0.00%
1	Ray	None	3	the stock exchange dipped	the stock exchange dipped	the-stock-exchange-dipped	21.308	0	25	25	13.51605031	0.00%
1	Ray	None	3	this is a very good idea	this is a very good idea	this-is-a-very-good-idea	17.0346	0	24	24	16.20231764	0.00%
1	Ray	None	3	so you think you deserve a raise	so you think you deserve a raise	so-you-think-you-deserve-a-raise	20.5906	0	32	32	18.66649636	0.00%
1	Ray	None	3	look in the syllabus for the course	look in the syllabus for the course	look-in-the-syllabus-for-the-course	26.7766	0	35	35	15.23718471	0.00%
1	Ray	None	3	protect your environment	protect your environment	protect-your-environment	14.0793	0	24	24	19.60324732	0.00%
1	Ray	None	3	be persistent to win a strike	be persistent to win a strike	be-persistent-to-win-a-strike	19.4224	0	29	29	17.29961282	0.00%
1	Ray	None	3	a big scratch on the tabletop	a big scratch on the tabletop	a-big-scratch-on-the-tabletop	20.7252	0	29	29	16.21214753	0.00%

Fig. 2. Sample log file for one participant covering 3 blocks with 8 phrases per block using one text entry technique using a particular modifier condition.

3.2 Text Entry Techniques

This subsection describes the main four text entry techniques used in our studies.

FLIK (Fluid Interaction Keyboard). One of our main contributions is the FLIK text entry technique. This text entry technique was inspired by CutieKeys [19]. CutieKeys employs a drumstick metaphor, where users enter text by swinging the VR controller like a drumstick to hit pads corresponding to keys organized in the standard QWERTY keyboard layout. However, unlike CutieKeys, FLIK supports contacting keys from both top and bottom directions. In our implementation of FLIK, the virtual keyboard presents keys as spheres, with each positioned far enough from its neighbor to prevent overlap or mistaken keystrokes due to accidental collisions between the cursor and the key sphere. FLIK uses bimanual 3D tracked controllers. A selection cursor sphere is attached to the tip of each controller; these selection spheres are what the user manipulates, and intersects with the key spheres to issue a specific keystroke. Intersection between the selection cursor and the key sphere simultaneously selects *and* confirms the key input; no further button presses are required with FLIK. Upon intersecting a key sphere, the user can move in arcs in both direction, and loop back through the keyboard to the desired key. In contrast, CutieKeys requires that the user move the controller back (upwards) from below upon striking a key, then move back down *again* from above to strike the next key; FLIK effectively cuts the total movement to half that required by CutieKeys.

Although this difference between FLIK and CutieKeys is subtle, we anticipate that this minor difference will improve entry rates due to the continuous movement of the hands. In contrast, with CutieKeys, the user has to tap on a key, and then reverse the direction of movement to stop selecting the key. See Fig. 3.

Fig. 3. Fluid intersection keyboard (FLIK). (A) Shows the phrase to be transcribed before the user has entered any text. (B) The user selects the character 'n' by moving his right hand through it so that the red selection cursor touches and goes through the character as seen in (C) where the sphere cursor is now behind (and partially occluded by) the 'n' character key sphere. (D) Again, shows the same right hand having looped around the bottom of the keyboard and selecting the 'o' character key sphere from beneath. (E) As the right hand exits the character 'o', the left hand now moves through the spacebar to select a space character. (F) User continues in this manner selecting the next character.

Controller Pointing. Controller pointing requires two 6DOF controllers, each of which emits a selection ray. Participants can bimanually (using both controllers, one in each hand) remotely point at a virtual (i.e., soft) keyboard presented in front of them. Individual keystrokes are issued upon pressing the primary thumb button on the controller. Visual feedback is provided by changing the colour of the intersected key. Vibrotactile feedback

and auditory feedback (a “click” sound) are also provided when the button is pressed. Both rays pointing at the same key would not have any side-effects. See Fig. 4.

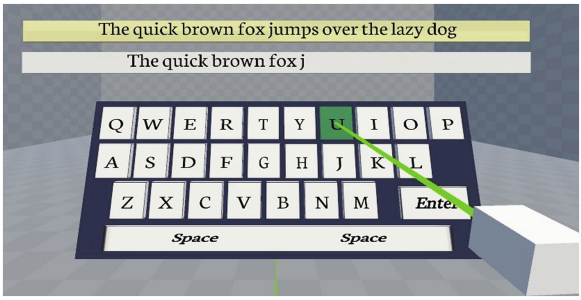

Fig. 4. Images showing the Controller Pointing keyboard

Continuous Cursor Selection. With continuous cursor selection, the virtual keyboard is divided in left and right halves to support bimanual entry. Keys on the right half of the keyboard are selected via a cursor controlled by the right hand and vice versa. A 2D selection cursor is initially placed at the midpoint on each keyboard half. These selection cursors are controlled by moving the corresponding (right or left) joystick (e.g., with an Oculus Touch controller) or sliding a finger/thumb on the controller’s touchpad (e.g., on an HTC Vive controller). The cursors move in the direction of joystick or touchpad movement. Once the cursor hovers over the desired key on the virtual keyboard, a button press on the corresponding controller confirms selection and issues the keystroke. The same feedback mechanisms described above for the controller pointing technique are employed with this technique as well. See Fig. 5.

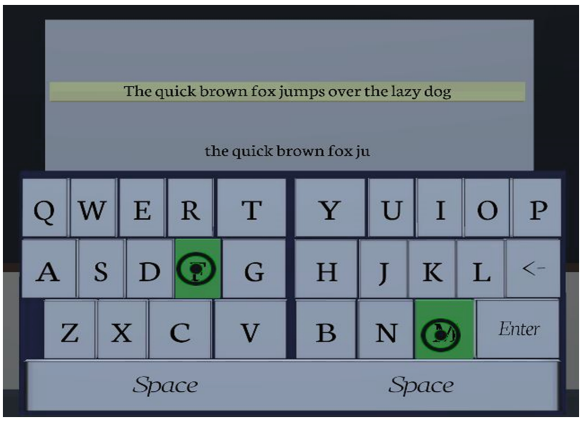

Fig. 5. Continuous Cursor Selection. By dragging each thumb on their respective touchpads, the cursors move on their respective half in order to select characters from the virtual keyboard.

Physical Keyboard. Lastly, as a baseline condition, we included a tracked physical keyboard with virtual hand representations via the Logitech Bridge API [18] and an HTC Vive individual tracker mounted on a keyboard. The hand representations are provided by the Logitech Bridge API via the HTC Vive's integrated camera and Logitech's proprietary computer-vision hand recognition techniques. This technique behaves the same as using a physical keyboard except that the user sees a virtual representation of the keyboard (and a video image of their hands) rather than the physical keyboard itself (Fig. 6).

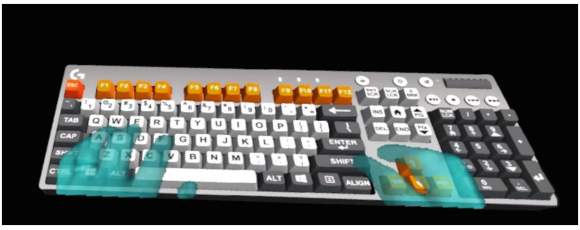

Fig. 6. Logitech Bridge virtual keyboard with hand representation

3.3 Text Entry Aids

The testbed supports the following add-on features that can be applied with any of the above text entry techniques (or any other that could be added). This demonstrates the flexibility of the testbed in providing a way to add different techniques with varying levels of complexity. The testbed currently supports two such aids, BigKey and word disambiguation, described below.

BigKey: BigKey [4] employs a custom algorithm which analyzes the input stream of the current word. The higher the probability a letter will be typed next, the bigger the size of its key on the virtual keyboard. The algorithm starts when an initial character for a new word is entered (i.e., following press of the space key). Prior to entry of the initial character for a new word, all keys are the same average size.

Following the first keystroke for the new word, the algorithm searches the phrase set for all possible words that start with entered character. Since the testbed included only a fixed phrase set size, there are a limited number of possible words that can be entered, so the overall computing time for finding matches is short. Prior to entering the next character, the system calculates the frequency of all subsequent next characters. This is done by counting the occurrences of each character in sequential order, starting with 'a', 'b', 'c', and so on, that directly follow the entered character for all words in the array. After this calculation is complete, all letters with 0 frequency (i.e., those that *never* follow the initial character) are scaled to the smallest size, while the letter or letters with highest frequency, are scaled up to the maximum size. Letters with frequencies between 0 and the highest frequency are scaled by a factor multiplier which starts with the scaling

factor of the letter with the highest count. The highest ranked letter would be scaled up by a factor of 1.75, the second by a factor of 1.7, and so on in decrements of 0.05. This ensures that all characters that *might* follow the initial character receive some scaling multiplier, with the more probable ones being scaled proportionally larger. See Fig. 7.

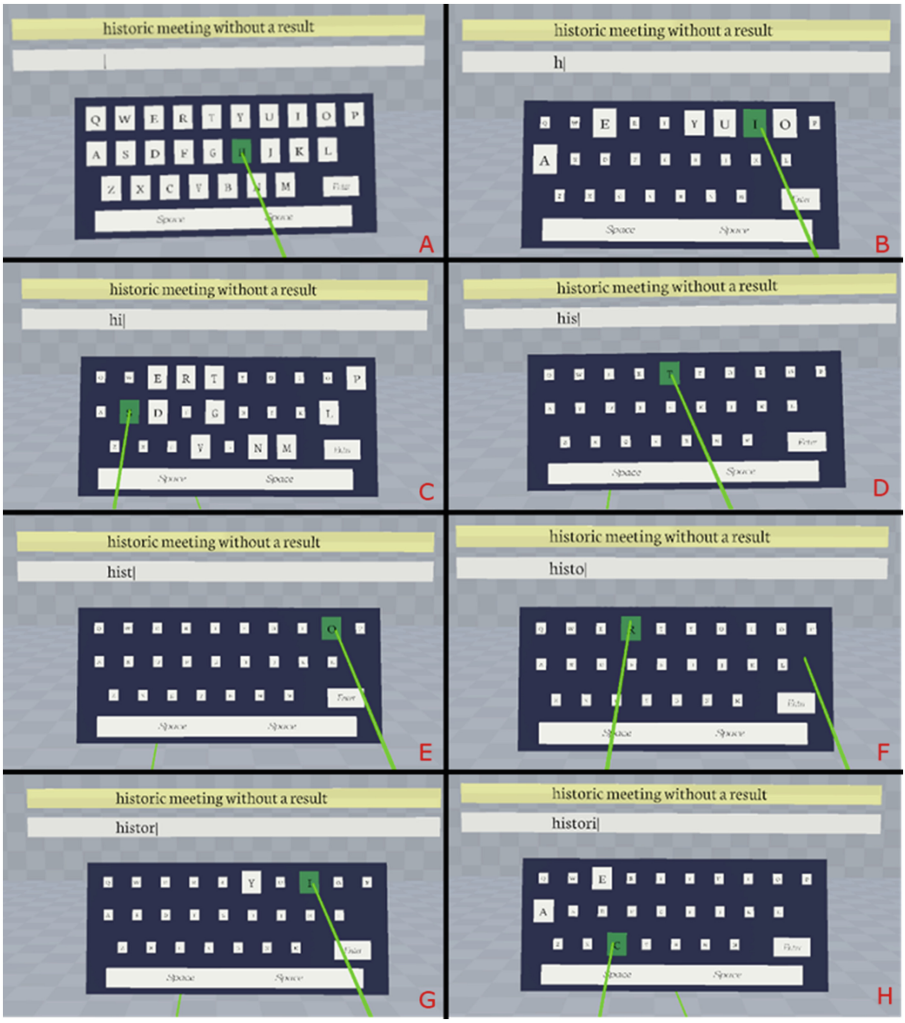

Fig. 7. Image showing keys rescaling on virtual keyboard based on BigKey algorithm. In every slide, a new character is selected, and the rest of the characters are resized based on the words found with those characters as an option to be typed next.

Word Disambiguation: In every language, there are frequencies of how often each word appears. From these frequencies, one can implement a general Word Disambiguation system that can be used with most text entry scenarios. The testbed implements this

in the form of suggestions, every time the user enters a key, the system computes the most likely word to be typed in order of ranking, then it displays the top three ranked words as selection options to complete a word in one selection rather than finishing the whole word.

This system works in a similar way to how key prediction is implemented for BigKey. The main difference is that with disambiguation, the ranking for every word in the phrase set is used. Along with Mackenzie's phrase set, a list of word rankings is provided. This list contains all words used in the phrase set along with their ranking. The rankings are in the form of a simple integer number, representing the frequency of use of this word.

4 User Study 1

This section presents the first user study using the text entry testbed comparing four VR text entry techniques. This first experiment consists of a performance comparison of the four text entry techniques for VR: Controller Pointing, Continuous Cursor Selection, FLIK, and Physical Keyboard.

4.1 Participants

This study included 24 participants, 14 male, 10 female, and aged 18–30 ($SD = 2.96$). All participants stated that they are comfortable with basic or advanced computer use.

4.2 Hardware

The experiment was conducted using a VR-ready laptop with an Intel core i7-6700HQ quad core processor, an Nvidia Geforce GTX 1070 GPU, and 16 GB of RAM, running the latest build of Microsoft Windows 10. The Logitech G810 was used as the physical keyboard due to integration requirement with the Logitech Bridge SDK. We used the HTC Vive HMD with its two touchpad-enabled 6DOF controllers. The HTC Vive provides a 1080×1200-pixel resolution per eye, 90 Hz refresh rate, and 110 degrees of field of view.

4.3 Software

The study employed the text entry testbed described above. It was developed in Unity 3D and it integrates easily with any text entry technique. The study included four text entry techniques: Controller Pointing, Continuous Cursor Selection, FLIK, and Physical Keyboard. Each text entry technique used in this study was developed in Unity independent of the text entry testbed and was then integrated with the testbed in order to use the full functionality of the system.

4.4 Procedure

Upon arrival, participants were informed of their right to withdraw from the experiment at any point without any obligation to finish the experiment if at any time they felt uncomfortable or nauseous. Participants were then asked to try the HMD, and were presented with their first text entry technique (based on counterbalancing order). They were given around two minutes to practice with the technique, to become familiar with the general system and the first text entry technique they would use.

The task involved transcribing presented phrases using the current text entry technique. Participants performed 3 blocks with each of the four text entry techniques, where each block consisted of 8 phrases to be transcribed. When all three blocks were finished, the next text entry technique was chosen and the process repeated. Participants were instructed to transcribe the phrases presented as quickly and accurately as possible.

4.5 Design

The experiment employed a within-subjects design, with two independent variables, text entry technique with four levels (FLIK, controller pointing, physical keyboard, and continuous cursor selection) and block with three levels (block 1, 2, and 3). The order of text entry technique was counterbalanced using a Latin square.

We recorded three dependent variables (entry speed, error rate, and NASA-TLX scores). Entry speed was measured in words per minute (WPM), and calculated as seen in Eq. 1. Error rate was based on the MSD, see Eq. 2.

$$Error\ Rate\ \% = \frac{100 \times MSD(P, T)}{\max(|P|, |T|)} \tag{2}$$

TLX scores are the overall results from the NASA-TLX questionnaire. Across all 24 participants, this yielded *3 blocks* $\times$ *8 phrases* $\times$ *4 text entry techniques* $\times$ *24 participants* $=$ *2304 phrases transcribed*.

4.6 Results

Task Performance. We first present quantitative results in terms of task performance, computed per participant, for each text entry technique, and averaged per block. Entry speed ranged from 12.32 WPM for Continuous Cursor Selection to 49.56 WPM for Physical Keyboard. See Fig. 8 for full results.

Repeated-measures analysis of variance revealed that the effect of text entry technique on entry speed was statistically significant ($F_{3,\,69} = 54.886$, $p < .0001$), as was the effect of block on entry speed ($F_{2,\,46} = 88.432$, $p < .0001$). A Post Hoc Bonferroni-Dunn test at the $p < .05$ level revealed pairwise significant differences between some text entry techniques, represented as horizontal bars on Fig. 8.

Error rates ranged from 1.83% error for physical keyboard to 4.03% for continuous cursor selection. Analysis of variance revealed that the effect of text entry technique on error rate was not statistically significant ($F_{3,\,69} = 0.431$, ns), nor was the effect of block ($F_{2,\,46} = 1.681$, $p > .05$). See Fig. 9.

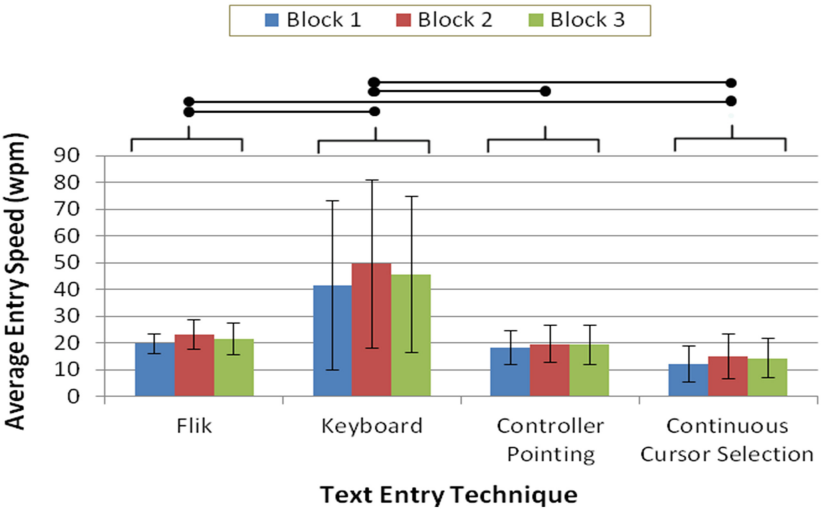

Fig. 8. Text entry speed average comparison chart. Error bars show standard deviation. Black horizontal bars show pairwise significant differences between text entry techniques.

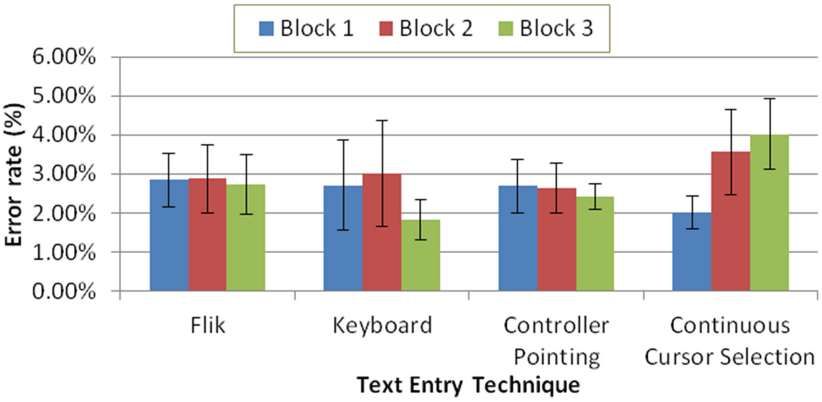

Fig. 9. Average error rate (%) comparison chart, error bars show standard deviation.

User Preference. User workload was assessed using Hart and Staveland's NASA Task Load Index (TLX). NASA TLX. A lower overall score signifies less workload and thus greater overall preference to a technique. Results were analyzed using a Friedman test, with Conover's test as a posthoc. Significant pairwise differences ($p < .05$) are represented on Fig. 10 by horizontal bars between each text entry technique.

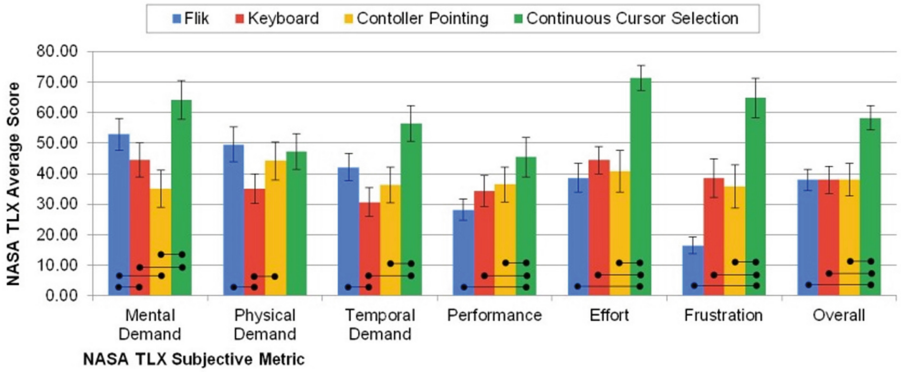

Fig. 10. Average results of NASA TLX questionnaire with standard error bars. Lower scores represent more positive results. Black bars depict significant pairwise differences.

4.7 Discussion

The physical keyboard condition did best overall, however, this was expected. After that, FLIK did about as well as controller pointing, with an average entry speed of 23 WPM on the third block, where the top entry speed achieved on any block was 28 WPM. When it comes to user preference and workload, FLIK scored worse on mental and physical demand. This is reasonable since this technique is similar to playing drums, where focus is placed on managing the accurate movement of hands as well as the physical effort of moving both hands continuously. Despite this, FLIK makes up for this in performance and low frustration, which are important factors when it comes to user preference. Due to the fact that these text entry techniques are, for the time being, meant to be used in short text entry tasks, the physical demand aspect becomes less of an issue since users would not be writing long form text, hence, reducing fatigue. Ranking first overall as participants' favorite text entry technique, it serves as proof that standard VR text entry techniques such as controller pointing, and continuous cursor can be outperformed and potentially replaced by more performant and preferred alternatives.

Physical keyboard was fastest, with a top entry speed score of 49.6 WPM on the third block. While there were some cases of low scores among some of the participants, with one participant scoring the lowest of 15.7 WPM, this is far offset with the proficiency of participants using regular keyboards, with most being able to type without looking at the keyboard. Cases where scores were low could be attributed to poor tracking, appearing as the virtual hands not aligning perfectly with real world hands. Such occurrences were rare and the mismatch between the virtual and real hands was not extreme. While the physical keyboard offered performance double that of the other techniques, we again note that the physical keyboard is, in many ways, a sub-optimal choice as a VR text entry technique due to several issues unrelated to performance. For example, the extra hardware (specific Logitech keyboard, attachment piece for the keyboard, and external HTC Vive sensors) and software required is not readily available with out of the box VR headsets. This decreases the adoption rate of such a technique, at least until these components become an integrated part of a HMD package. The most negative factor is

that physical keyboards are not very portable. VR users tend to be standing and walking, rotating in place, and generally changing their position more often than not. Carrying a full-sized keyboard does not lend itself very well for this type of activity since it would need to be stationary.

Controller pointing is currently the most commonly used technique in commercial VR products, and was shown to offer good performance as well. However, with text entry speed results coming in below FLIK (although not significantly worse), and being overall ranked second as participants' favorite choice of text entry technique, there is room for improvement when it comes to the standard method of entering text in VR. The continuous cursor technique, which is also commonly available in commercial products, was by far the least preferred and worst performing technique, further demonstrating that the industry VR software developers have gravitated towards several sub-optimal techniques.

5 User Study 2

We present a second user study comparing performance between Controller Pointing and FLIK with the addition of text entry aids described earlier, BigKey and word disambiguation. To further demonstrate the flexibility of the text entry testbed, we used an Oculus Rift CV1 in this study instead of the HTC Vive, which have different setup requirements and APIs.

5.1 Participants

This second study consisted of 24 participants, 13 male, 11 female, and aged 18–55 (*SD* = 9.01). None of these participants completed the first user study.

5.2 Hardware

The experiment employed same hardware as the first study, with the exception of using an Oculus Rift CV1 HMD with two Oculus Touch 6DOF controllers instead of the HTC Vive. The Oculus Rift CV1 provides a 1080 × 1200-pixel resolution per eye, a 90 Hz refresh rate, and 110 degrees of field of view.

5.3 Software

The text entry testbed was again used for this study. We used a subset of the text entry techniques from the previous study. The only new additions to the software were the BigKey and word disambiguation aids.

5.4 Procedure

Upon arrival, participants were told that if at any time they felt uncomfortable or nauseous that they were free to stop the experiment without finishing. Participants were then asked to try the HMD, followed by presenting them with their first text entry technique with

the first condition (no aids, BigKey, or word disambiguation), where they were free to try it out and practice for around two minutes. Once they felt comfortable, the study began with that text entry technique.

This study employs the same procedure as the first study, but with the techniques and aids described below.

5.5 Design

The experiment employed a within-subjects design, with three independent variables:

Text entry technique: FLIK, controller pointing.
Aid: no aid, BigKey, Word Disambiguation.
Block: 1, 2, and 3.

The text entry techniques and aids were counterbalanced using a Latin square. With all 24 participants, this resulted in *3 blocks * 8 phrases * 2 text entry techniques * 3 aids * 24 participants = 3456 phrases transcribed.*

5.6 Results

Task Performance. The slowest condition was Controller Pointing with no aid, at 18.12 WPM. The fastest condition, at 27.8 WPM was FLIK+BigKey. See Fig. 11.

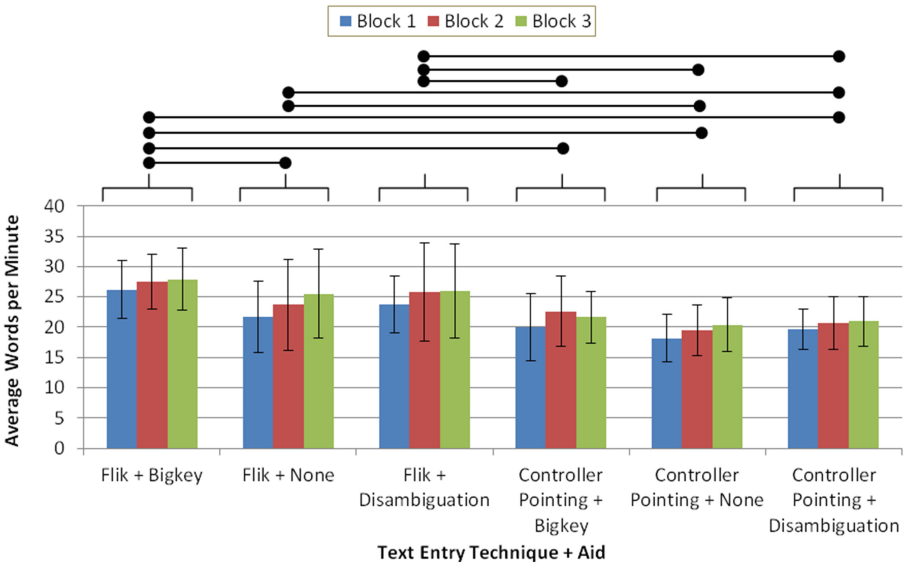

Fig. 11. Study 2 average entry speed comparison table. Error bars show standard deviation. Black horizontal bars show pairwise significant differences between some text entry techniques

Repeated-measures ANOVA revealed that the effect of text entry technique on entry speed was statistically significant ($F_{1, 23} = 67.705$, $p < .0001$), as was aid ($F_{2, 46} = 41.098$, $p < .0001$) and block ($F_{2, 46} = 107.446$, $p < .0001$). The interaction effects for text entry technique-block ($F_{2, 46} = 5.338$, $p < .01$) and aid-block were also statistically significant ($F_{4, 92} = 6.179$, $p < .0005$). Post Hoc Bonferroni-Dunn test at the $p < .05$ level results are represented as horizontal bars showing individual significant results for each text entry technique + aid combination.

Error rates ranged from 0.73% error for Controller Pointing + BigKey to 4.11% for FLIK + word disambiguation. Repeated-measures analysis of variance revealed that the effect of text entry technique on error rate was statistically significant ($F_{1, 23} = 6.456$, $p < .05$). The effect of aid on error rate was statistically significant ($F_{2, 46} = 23.412$, $p < .0001$). The text entry technique-block interaction effect was statistically significant ($F_{2, 46} = 13.855$, $p < .0001$). The aid-block interaction effect was significant ($F_{4, 92} = 12.067$, $p < .0001$). A post hoc Bonferroni-Dunn test at the $p < .05$ level showed individual significant results on error rate for each of the text entry technique + aid combination and are represented by horizontal bars on Fig. 12.

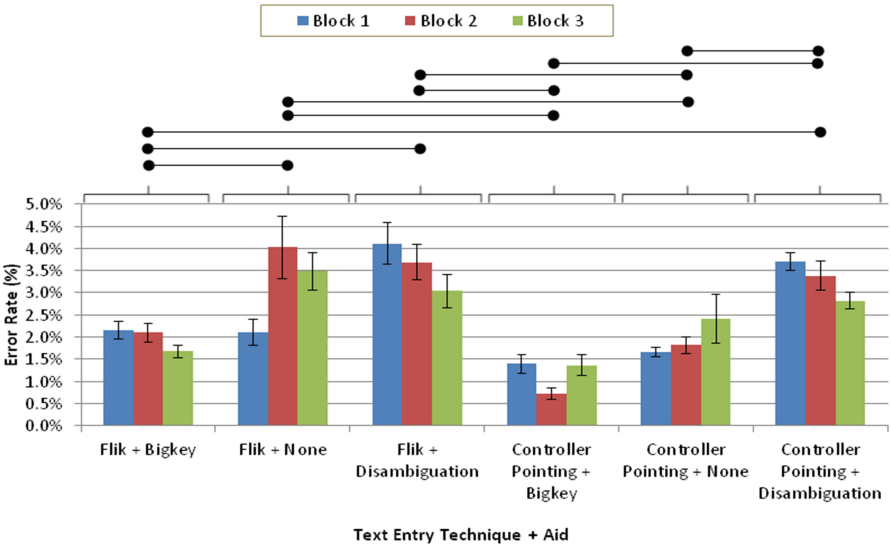

Fig. 12. Study 2 average error rate (%) comparison table. Black horizontal bars show pairwise significant differences between some text entry techniques.

User Preference. As seen in Fig. 13, FLIK + None and Controller Pointing + None scored similarly to corresponding conditions in study 1. Friedman post hoc pairwise comparison using Conover's F test results are represented by horizontal bars between each text entry technique.

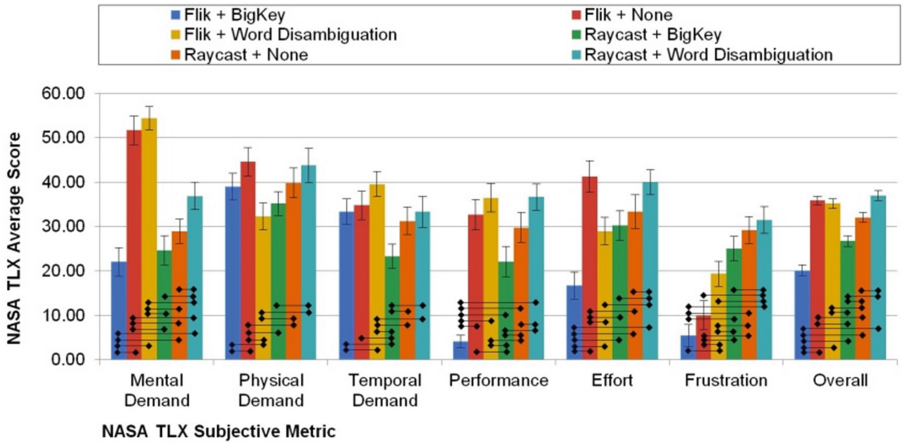

Fig. 13. Averages of study 2 NASA TLX results

5.7 Discussion

Using BigKey with FLIK improved both performance and satisfaction, yielding the highest entry speed at 27.9 WPM. This suggests that there is potential in alternative VR text entry techniques, and how they can be improved with different tweaks and aids, even offering better performance than common commercial approaches, such as controller pointing. Word disambiguation offered poor results and actually caused some frustration and higher mental demand than other text entry technique + aid combinations. BigKey offered the best overall performance with FLIK.

Given the error rates and entry speeds observed, we speculate that choosing and selecting fully suggested words from the suggestion list increases mental workload, particularly since VR text input is already something new to most people. On the other hand, BigKey is non-intrusive and works with the participant in the task of purely typing character by character, making it easier and more obvious to type the next character in the sequence, achieving a greater flow in the typing task as well as user satisfaction. The higher performance of BigKey can be attributed to Fitts' law, which states that the time required to rapidly move to a target area is a function of the ratio between the distance to the target and the width of the target.

6 Conclusion and Future Work

We developed a general-purpose text entry testbed focused on VR text entry experiments. Using the testbed, we conducted two comparative studies that evaluate different text entry techniques.

Our results for controller pointing and for continuous cursor selection are comparable to results provided by Speicher et al. [17]. While our scores for controller pointing were slightly higher than either Speicher et al. [17] or Boletsis and Kongsvik [3], our results reflect the same relative difference between controller pointing and continuous cursor selection. Boletsis and Kongsvik [2] evaluation of CutieKeys, which is the basis of our

technique FLIK, reveals similar performance between the two techniques, and the same relative difference between these techniques and others. Table 1 summarizes these results for easy comparison.

Table 1. Text entry speed (WPM) comparison to similar studies.

	Our studies	Speicher [17]	Boletsis [2]	Grubert [6]	Knierim [7]
Controller pointing	19.13	15.44	16.65	–	–
Continuous cursor selection	13.9	8.35	10.17	–	–
FLIK	21.49	–	*21.01	–	–
Physical keyboard	45.64	–	–	38.7	40

*Used CutieKeys technique, which is the most similar to Flik.

Grubert et al. [6] reported an entry speed of 38.7 WPM text entry speed on their VideoHand technique, which is what our physical keyboard technique is based on. Table 1 again shows that these related papers report similar results on keyboard entry speeds in VR. These papers also report high standard deviation with keyboard-based input, likely due to including participants with varying typing proficiency, like our studies.

For future iterations of the testbed, a main focus is to improve extensibility, to require less programming ability to add new text-input techniques and/or keyboard layouts. This could use, for example, XML-based approaches to define new techniques/layouts.

Other future work could explore different kinds of text entry experiments using this software. From plain desktop text entry techniques to virtual or augmented reality techniques, as well as other types of human-computer interactions other than these such as wearable devices (smart watches, smart glasses…), brain-computer technologies such as EEG's like EMOTIV devices [3]. Conducting studies comparing different types of HMDs and controllers is also an important study to perform since this could potentially be a significant factor when it comes to text entry in VR.

Acknowledgments. Thanks to all participants for taking part in the experiments. Thanks to Aidan Kehoe of Logitech for providing the Logitech Bridge hardware. This work was supported by the Natural Sciences and Engineering Research Council of Canada.

References

1. Arif, A.S., Stuerzlinger, W.: Analysis of text entry performance metrics. In: IEEE Toronto International Conference on Science and Technology for Humanity (TIC-STH), pp. 100–105 (2009). https://doi.org/10.1109/tic-sth.2009.5444533
2. Boletsis, C., Kongsvik, S.: Controller-based text-input techniques for virtual reality: an empirical comparison. Int. J. Virtual Real. **19**(3), 2–15 (2019). https://doi.org/10.20870/ijvr.2019.19.3.2917
3. EMOTIV: EMOTIV. https://www.emotiv.com/. Accessed 10 Feb 2020

4. Al Faraj, K., Mojahid, M., Vigouroux, N.: BigKey: a virtual keyboard for mobile devices. In: Jacko, J.A. (ed.) HCI 2009. LNCS, vol. 5612, pp. 3–10. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02580-8_1
5. Grubert, J., Witzani, L., Ofek, E., Pahud, M., Kranz, M., Kristensson, P.O.: Text entry in immersive head-mounted display-based virtual reality using standard keyboards. In: Proceedings of IEEE Conference on Virtual Reality and 3D User Interfaces, pp. 159–166 (2018)
6. Grubert, J., Witzani, L., Ofek, E., Pahud, M., Kranz, M., Kristensson, P.O.: Effects of hand representations for typing in virtual reality. In: Proceedings of IEEE Conference on Virtual Reality and 3D User Interfaces, pp. 151–158 (2018)
7. Knierim, P., Schwind, V., Feit, A.M., Nieuwenhuizen, F., Henze, N.: Physical keyboards in virtual reality: analysis of typing performance and effects of avatar hands. In: Proceedings of ACM Conference on Human Factors in Computing Systems, pp. 1–9 (2018). https://doi.org/10.1145/3173574.3173919
8. Ko, A.J., Wobbrock, J.O.: Text entry. User Interface Software and Technology. https://faculty.washington.edu/ajko/books/uist/text-entry.html. Accessed 18 June 2020
9. MacKenzie, I.S.: Fitts' law. In: Norman, K.L., Kirakowski, J. (eds.) Handbook of Human-Computer Interaction, pp. 349–370. Wiley, Hoboken (2018). https://doi.org/10.1002/9781118976005
10. Mackenzie, I.S.: Evaluation of text entry techniques. In: MacKenzie, I.S., Tanaka-Ishii, K. (eds.) Text Entry Systems: Mobility, Accessibility, Universality, pp. 75–101 (2007). https://doi.org/10.1016/b978-012373591-1/50004-8
11. MacKenzie, I.S., Soukoreff, R.W.: Phrase sets for evaluating text entry techniques. In: Extended Abstracts on Human Factors in Computing Systems - CHI 2003, pp. 754–755 (2003). https://doi.org/10.1145/765891.765971
12. MacKenzie, I.S., Soukoreff, R.W.: A character-level error analysis technique for evaluating text entry methods. In: Proceedings of the Nordic Conference on Human-Computer Interaction - NordiCHI. 2002, p. 243 (2002). https://doi.org/10.1145/572020.572056
13. Sandnes, F.E., Aubert, A.: Bimanual text entry using game controllers: relying on users' spatial familiarity with QWERTY. Interact. Comput. **19**(2), 140–150 (2007). https://doi.org/10.1016/j.intcom.2006.08.003
14. Soukoreff, R.W., MacKenzie, I.S.: Measuring errors in text entry tasks. In: Extended Abstracts on Human Factors in Computing Systems - CHI 2001, pp. 319–320 (2001). https://doi.org/10.1145/634067.634256
15. Soukoreff, R.W., MacKenzie, I.S.: Recent developments in text-entry error rate measurement. In: Extended Abstracts on Human Factors in Computing Systems - CHI 2004, pp. 1425–1428 (2004). https://doi.org/10.1145/985921.986081
16. Soukoreff, R.W., MacKenzie, I.S.: Metrics for text entry research - an evaluation of MSD and KSPC, and a new unified error metric. In: Proceedings of ACM Conference on Human Factors in Computing Systems - CHI 2003, pp. 113–120 (2003). https://doi.org/10.1145/642611.642632
17. Speicher, M., Feit, A.M., Ziegler, P., Krüger, A.: Selection-based text entry in virtual reality. In: Proceedings of ACM Conference on Human Factors in Computing Systems - CHI 2018, pp. 1–13 (2018). https://doi.org/10.1145/3173574.3174221
18. Tucker, V.: Introducing the logitech BRIDGE SDK (2017). https://blog.vive.com/us/2017/11/02/introducing-the-logitech-bridge-sdk. Accessed 18 June 2020
19. Weisel, M.: Cutie keys. https://github.com/NormalVR/CutieKeys/. Accessed 18 June 2020
20. Wobbrock, J.O.: Measures of text entry performance. In: MacKenzie, I.S., Tanaka-Ishii, K. (eds.) Text Entry Systems: Mobility, Accessibility, Universality, pp. 47–74 (2007)