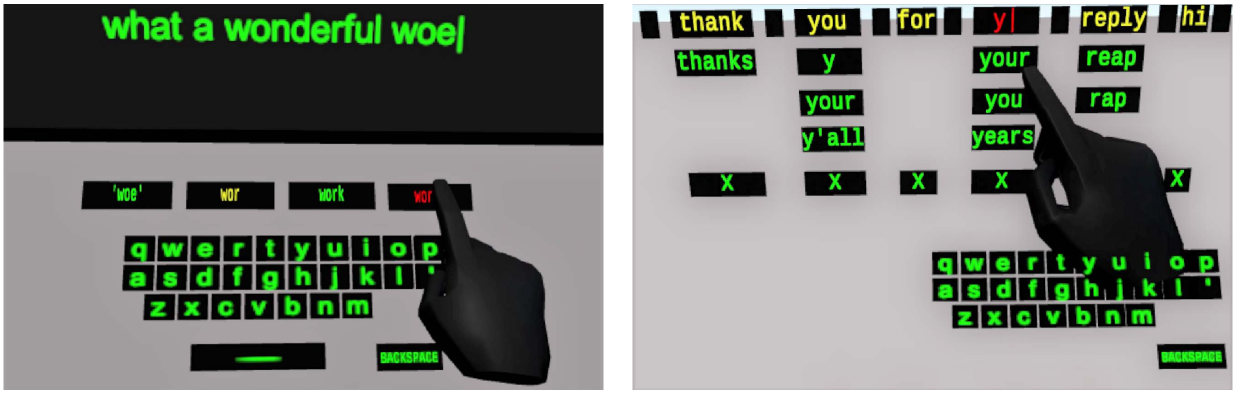
Text Entry in Virtual Environments using Speech and a Midair Keyboard

Jiban Adhikary Keith Vertanen



what a wonderful woe|

Fig. 1. We compare typing on a midair auto-correcting keyboard with word predictions (left) versus speaking a sentence and then correcting any speech recognition errors (right). Users correct errors by selecting word alternatives proposed by the speech recognizer or by typing on the virtual keyboard.

Abstract— Entering text in virtual environments can be challenging, especially without auxiliary input devices. We investigate text input in virtual reality using hand-tracking and speech. Our system visualizes users' hands in the virtual environment, allowing typing on an auto-correcting midair keyboard. It also supports speaking a sentence and then correcting errors by selecting alternative words proposed by a speech recognizer. We conducted a user study in which participants wrote sentences with and without speech. Using only the keyboard, users wrote at 11 words-per-minute at a 1.2% error rate. Speaking and correcting sentences was faster and more accurate at 28 words-per-minute and a 0.5% error rate. Participants achieved this performance despite half of sentences containing an uncommon out-of-vocabulary word (e.g. proper name). For sentences with only in-vocabulary words, performance using speech and midair keyboard corrections was faster at 36 words-per-minute with a low 0.3% error rate.

Index Terms—Text Entry, Speech Recognition, Virtual Reality (VR), Head Mounted Display (HMD), Midair Gestures.

1 Introduction

People frequently interact with text on desktop computers and mobile devices. With the rise of virtual reality (VR) and augmented reality (AR) head mounted displays (HMDs), there is a need to support efficient text interaction in virtual environments. However, text entry in virtual environments is entirely different from typing on a physical keyboard of a desktop computer or even on a virtual keyboard of a mobile device. This is due to different fields of view, display size, and lack of tactile feedback. Humans have a field of view of approximately 180 degrees, whereas the field of view in an HMD is limited and typically ranges from 90 to 110 degrees. Most importantly virtual environments lack the perception of touch when interacting with the virtual objects. HMDs are normally shipped with a hand-held controller as an input device. While a physical keyboard can be used, in many VR and AR use scenarios, users may be standing or moving around. Such use scenarios make use of auxiliary input devices such as hand-held controllers or physical keyboards difficult. Ideally input would be possible using just the built-in sensors common in HMDs such as depth cameras and microphones. We investigate exploiting these two types of sensors to support text input in virtual reality.

For modest amounts of input, we argue a familiar input method leveraging users' experience with auto-correcting touchscreen keyboards may be preferable. As only the VR or AR user can see the keyboard, a virtual keyboard also provides a reasonable degree of privacy. Further, with appropriate design, a virtual keyboard can provide entry of even difficult to predict text (e.g. proper names or passwords). However, typing in midair may be slower than on a touchscreen due to factors such as the lack of tactile feedback and hand-tracking inaccuracies. When privacy is not a concern, speech may be faster; people have been measured dictating to a computer at over 100 words-per-minute (wpm) [23]. However, speech recognition errors can occur, especially for difficult text or in noisy environments. Correcting speech recognition errors with speech is a possibility, but can lead to errors while trying to correct errors [20]. Switching to another input modality is a common approach to this problem, e.g. [32, 40]. In this paper, we investigate correcting errors via midair tapping. Our interface supports error correction by selecting alternative words proposed by the speech recognizer or via an auto-correcting keyboard that exploits the surrounding text context.

The main goal of our study is to investigate how we can incorporate speech recognition into a VR text entry interface. An additional goal is to support the input of more difficult text, e.g. proper names or passwords. We argue for an interface to be usable in the real-world, it needs to support at least occasional entry of such text, even if it requires slower and more precise input. To support the input of difficult text, we add word predictions to our keyboard, including the literal letters typed. Past work has shown such suggestions can enable the input of challenging text on a smartwatch [42]. Previous work has also reported

- *Jiban Adhikary (jiban@mtu.edu) and Keith Vertanen (vertanen@mtu.edu) are with the Department of Computer Science at Michigan Technological University.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

faster input when word predictions were added to an AR keyboard [10].

We provide the first study into text input in a VR HMD using speech and a midair virtual keyboard with hand interaction. We found speech was over twice as fast as an auto-correcting keyboard with word predictions (28 wpm versus 11 wpm). Speech also had a lower error rate (0.5% versus 1.2%). This was achieved on text with uncommon words. Our study adds to the limited work on hand tracking based keyboard input in virtual environments. It also adds to our knowledge about how to support the input of challenging text, including showing speech provides increased performance even on text often requiring corrections.

2 Related work

In this section, we overview existing work in text entry in virtual environments, with a focus on the input modalities employed in the interface. For a survey of existing work related to text entry in virtual environments, including a taxonomy of techniques and a discussion of non-QWERTY keyboard designs, see Dube and Arif [8].

2.1 Physical Keyboards

Various work has investigated VR input using a physical keyboard. McGill et al. [29] found users could type efficiently on a physical keyboard if real-world video was injected into VR. Walker et al. [49] combined a physical keyboard with auto-correction. Providing only visual feedback after each key press allowed typing at 40 wpm.

Knierim et al. [21] studied hand representation in VR with a physical keyboard. They found appropriate hand visualization was especially important for inexperienced typists. Grubert et al. [15] rendered a user's fingers in VR. Users typed at 26 wpm on a deterministic physical keyboard. Grubert et al. [14] also investigated different hand representations while typing on a physical keyboard. They found hand visualization did not affect entry rate but did lower error rate. Similarly, Otte et al. [31] showed typing on a touch-sensitive physical keyboard without finger visualization was as effective as with finger visualization.

Pham and Stuerzlinger [33] introduced HawKEY, a physical keyboard worn on a tray suspended in front of the user. They mounted a color and depth sensing camera on the HMD and users typed on a physical keyboard with different visualizations. They found a see-through video condition was superior to four other approaches: no visualization, visualization of the keyboard frame, a virtual keyboard matching the dimensions and appearance of the physical keyboard, and a point cloud representation of a user's hand and the keyboard.

ReconViguRation [36] investigated several methods to reconfigure the presentation of the keyboard in the virtual environment. In a user study, they tested nine VR-relevant applications with the reconfigured design. Users reported that the applications were usable. For a password entry application, they found shuffling the keys in a local region of the keyboard was enough for secure password entry.

2.2 Hand-held Controllers

Other work has investigated input using hand-held controllers and a virtual keyboard rendered in an HMD. Yu et al. [55] combined head-tracking with a gamepad controller. A word-gesture keyboard [57] approach was the fastest at 25 wpm after an hour of practice. Xu et al. [52] compared text input on an AR keyboard using four pointing methods: controller, head, hand, and hybrid (head pointing and hand selection) and two input techniques: word-gesture input and letter-tap input. They found that using a controller was the best in terms of text entry performance and user experience. They also found that word-gesture input is as fast as letter-tap even for users who are new to word-gesture technique.

Speicher et al. [38] investigated six selection-based methods for VR text entry. Users typed on a virtual keyboard shown in a head mounted display. Users performed input by hand-held controllers or by midair gestures with hand visualization provided by a Leap Motion sensor. They found hand-held controller pointing outperformed other methods with an entry rate of 15 wpm at a 1% error rate. The hand visualization method had an entry rate of 10 wpm with a much higher error rate of 7.6%. We used a similar hand visualization approach.

Boletsis et al. [3] compared four hand-held controller-based VR text-input techniques: pointing using a controller, hitting keys on a drum-like virtual keyboard by making downward strikes with the controller, pointing at a key using head rotation and selecting with the controller, and a virtual keyboard split between the trackpads on the left and right controllers. They found controller pointing and striking was the fastest at 21 wpm but was the least accurate with a total error rate of 12%.

Xu et al. [53] proposed hands-free text entry on a circular keyboard rendered in an HMD and operated via head motion. With 60–90 minutes of training, users wrote at 11–13 wpm. Jiang et al. [19] explored text entry in VR by leveraging the circular touchpad of a HTC Vive controller and a circular virtual keyboard. They investigated three different key layouts and found a 6-key layout was best. They optimized the layout based on touch data and investigated whether to display only the optimized keyboard layout in the virtual environment or on top of the touchpad of a virtual representation of the controller. Using one hand, users entered text at 13.6 wpm with the virtual layout, and 11.6 wpm with the touchpad of the virtual controller.

2.3 Hand Gestures

A number of studies have exploited hand or finger gestures for VR/AR input. Markussen et al. [27] tested three selection-based approaches with input sensed via a tracked glove. A deterministic keyboard was the fastest at 13 wpm after four hours of practice. In Vulture [28], users also wore a glove and wrote via a word-gesture keyboard [57]. After five hours of practice, users reached 20 wpm.

HoldBoard [1] used a smartwatch for input with results displayed on smartglasses. A combination of thumb position and tapping allowed deterministic character entry. After eight sessions, users wrote at 10 wpm. In Yu et al. [56], users wrote at 9 wpm using one-dimensional touch input and auto-correction on a Google Glass HMD. ARKB [24] used a stereo visible light camera to track colored markers on a user's fingers and detected taps on a virtual keyboard. No user trial was reported. PalmType [50] displayed a keyboard on a user's palm via a Google Glass HMD. Users typed at 8 wpm using a Vicon tracking system and 5 wpm using a wrist-worn infrared sensor.

Similar to our work, Sridhar et al. [39] and Feit et al. [11] used finger input sensed via a Leap Motion. However, both required users to learn specific multi-finger gestures. Our approach allows users to reuse their existing experience with auto-correcting touchscreen keyboards.

The ATK system [54] allowed two-handed 10 finger touch typing in midair with visual feedback on a desktop display. While potentially fast, ATK currently relies exclusively on probabilistic entry and thus would not work well for difficult text, e.g. passwords. ATK also placed the Leap Motion under a user's hands to minimize occlusion. The sensor was stationary and required users to register a home hand position. Compared to ATK, we support the input of difficult text via a literal prediction slot. Our approach also ensures users can freely move their hands and head by mounting the hand tracker on the HMD. This freedom of movement would make ATK challenging to use while standing or walking.

VISAR [10] is an AR midair auto-correcting keyboard. Similar to our keyboard, it uses the VelociTap decoder [48]. VISAR was tested on a Microsoft HoloLens HMD that tracked a user's wrist location. Without word predictions, users wrote with one hand at 6 wpm. With word predictions, users wrote faster at 18 wpm after two hours of practice. Compared to VISAR, our system tracks a user's individual fingers and supports speech input. Dudley et al. [9] explored typing in VR with index fingers on a surface and in midair, and typing using all ten fingers on a surface and in midair. They found users typed faster on a virtual keyboard if it was aligned with a physical surface. They also found typing on a midair keyboard with ten fingers was actually slower and less accurate compared to typing with just two index fingers.

2.4 Speech

Only a handful of studies have investigated using speech for text input in virtual environments. Bowman et al. [6] compared VR text input using a chorded keyboard, Wizard-of-Oz speech recognition, and a tablet keyboard. In the speech condition, users spelled each word. Speech was the fastest at 13 wpm.

In SWIFTER [34], users spoke a sentence and then used a hand-held controller to click targets in a correction interface. The interface showed the recognized sentence along with markers between words. Clicking on a word opened a list of alternatives for that word. Clicking a marker allowed insertion between words. New words were entered by speaking the word or by speaking its spelling. In a CAVE virtual environment, users wrote German phrases at 24 wpm at a low error rate. Our work differs in the following ways: 1) we use a VR HMD for display instead of a CAVE, 2) we allow direct hand interaction with the interface rather than via a controller, 3) we use a virtual keyboard to correct errors rather than speech, and 4) we immediately display alternatives for each word rather than requiring explicit selection.

Similar to our system, SpeeG [17] and SpeeG2 [18] relied just on speech and hand gestures. Both SpeeG and SpeeG2 used a large screen for display with gestures detected via a Kinect. In SpeeG, users spoke a sentence and used hand gestures to zoom character-by-character through the words proposed by the recognizer. Users entered text using SpeeG at 11 wpm. SpeeG2 used a word-at-a-time correction interface employing gestures to select word alternatives. Similar to SWIFTER, words can be inserted between words and new words can be added via spelling. The best SpeeG2 prototype had an entry rate of 21 wpm.

Our speech correction interface is based on a word confusion network (WCN) [26]. A WCN is a time-ordered set of clusters where each cluster contains word hypotheses and their probabilities. Our design is similar to the Parakeet interface [44]. Parakeet allowed users to speak a sentence and then select word alternatives by tapping or swiping on a touchscreen mobile device. Users wrote at 18 wpm while seated indoors and 13 wpm while walking outdoors. Similar WCN speech interfaces have been used in desktop [30] and large display [22] environments. Compared to past WCN interfaces, our work differs in a number of ways. First, we use an auto-correcting virtual keyboard that leverages the text both to the left and to the right of the correction. Parakeet had a keyboard without auto-correct and its word prefix predictions used a unigram language model that ignored surrounding text. Second, we show a WCN correction interface is effective even with hand tracking which is substantially noisier than touchscreen or pen input. Further, in our VR study users could only see a virtual and noisy representation of their input device (i.e. their hands). Finally, we investigate the input of difficult to predict text by adding a literal prediction slot.

In recent years, speech recognition accuracy has markedly improved [16]. Limited work has investigated input using modern recognizers. In Ruan et al. [35], users input text at 153 wpm using the Baidu recognizer. Error correction was done on a phone using speech or a virtual keyboard. We also use a modern recognizer, namely IBM Watson. While our entry rates are slower, we evaluated using more difficult text that often required some correction. Additionally, even for perfect recognition results, our entry rate includes the time users spent reading and confirming results.

Foley et al. [12] investigated speech recognition for both the transcription and composition of text. Using the default keyboard and speech recognizer of a Google Pixel 3, users composed at 117 wpm using speech versus 35 wpm using the keyboard. They transcribed text at 157 wpm using speech versus 48 wpm using the keyboard . Overall, the entry rate of sentences with corrections were slower and less accurate at 55 wpm and 0.65% uncorrected error rate [37] versus sentences without corrections at 141 wpm and 0.52% uncorrected error rate.

To summarize, compared with past work, we focus on designing an input method that does not require training, hand-held devices, gloves, a physical keyboard, or expensive tracking infrastructure. Further, we investigate how to allow controller-less correction of speech recognition results. Finally, most past interfaces were evaluated on fairly easy text, e.g. the MacKenize [25] or Enron [46] phrase sets. We aim to design an interface supporting the input of more challenging text.

3 Interface Design

In this section, we detail the features of our midair keyboard and explain our design choices. Our interface supports input using two techniques: 1) using a virtually rendered QWERTY keyboard in midair, and 2) using speech with the virtual keyboard as a fallback mechanism.

3.1 Midair Auto-correcting QWERTY Keyboard

At least at present, many VR/AR applications focus on immersive 3D experiences such as games, training simulations, and data visualization. With advances in hand tracking and speech recognition, such applications can often be controlled without input devices aside from the HMD itself. This has the potential for more natural, convenient, and immersive interactive experiences. But even these applications may benefit from occasional input of text (e.g. short text messaging or executing a search query). The QWERTY keyboard is the de facto standard for English text input on desktop and mobile devices. This makes a midair QWERTY virtual keyboard an obvious choice as it is familiar to most users. Further, virtual keyboard decoding is quite accurate even without per-user training (though such training can provide small accuracy gains, e.g. [51]).

We rendered the QWERTY keyboard in front of a user (Figure 1) and anchored the keyboard in space. The keyboard has the letters A–Z plus apostrophe. The keyboard size is 21.0 × 7.6 cm with each letter key being 1.8 × 1.4 cm. After each key press, the nearest key label is highlighted in red, a tap sound is played, and the letter is added to the text above the keyboard. Users can remove previous characters using a backspace key of size 6.0 × 1.4 cm. We made the backspace key bigger and located it away from other parts of the keyboard to allow easy and deterministic triggering. Letters from the current word as well as previous words can be backspaced.

The space key is 8.5 × 1.4 cm. Triggering the space key results in changing the currently typed letter sequence into the most probable word given a user's sequence of noisy (x, y) tap locations on the keyboard plane. Similar to backspace, we made the space key bigger and separated it from the rest of the keyboard to allow deterministic triggering. For auto-correction, we use the VelociTap decoder [48]. VelociTap takes a sequence of noisy taps and searches for the most probable word using a probabilistic keyboard model, a character language model, and a word language model. The likelihood of a tap is based on a two-dimensional Gaussian centered at each key. For each tap, the decoder computes the likelihood for every key under the keyboard model. This is added with the language model probabilities. The language models condition on any text to the right and left of the current typing location. We used a 12-gram character language model and a 4-gram 100K word language model. Both language models were trained on billions of words of text. The set of parameters controlling VelociTap's operation were optimized on four users not in our user study.

In testing by the authors and two participants not in our user study, we found it was time consuming to replace a misrecognized word by repeatedly tapping backspace. We added a delete-word key that deleted the entire previous word. This key appeared to the right of the backspace key after a user typed a space. As with space, and backspace, we made this key bigger (4.0 × 1.4 cm) to make it easier to tap.

We chose a midair vertical keyboard for our pilot. We found the distance worked well, but based on user feedback, we tilted the keyboard approximately 10 degrees from vertical and lowered the keyboard to slightly below the user's chest level. This allowed users to see the keyboard and input progression with minimal head movement. While a horizontal keyboard aligned with a surface, such as a table, might more closely resemble desktop typing, hand tracking inaccuracies, virtual keyboard rendering inaccuracies, and hunt-and-peck typists may require visual supervision of motor actions over the keyboard. This would require frequent changes in head position to shift attention between the keyboard and any visual content in front of the user. Finally, even if users could touch type while looking straight ahead, this requires an HMD supporting visual rendering and hand tracking at quite a wide angle of view.

3.2 Support for Difficult Text

Text entry evaluation studies in virtual environments have mostly used phrases which are easy to remember [25, 46]. Such phrases have been shown to exhibit a low rate of out-of-vocabulary words and a low perplexity under a language model [42]. This allows users to enter such phrases quite accurately even despite severe inaccuracies in a user's tap locations. For example, in a study on a smartwatch keyboard [41] participants' character error rate was 19% before auto-correction, but only 3% after auto-correction. However, auto-correction may not be as effective for difficult and unpredictable text, e.g. proper nouns, passwords, or abbreviations. Such words are problematic for entirely recognition based input approaches such as gesture keyboards or continuous speech recognition. In our design, we support input of difficult words by relying on users' ability to carefully target keys when necessary combined with an interface option that avoids auto-correction (to be discussed shortly). At least for noisy input on a smartwatch, it was shown that users could precisely target every character of difficult words [42]. We conjectured a similar approach would be effective for noisy midair keyboard input.

We opted for midair keyboard input of difficult words rather than spoken spelling of words. When typing words character-by-character in midair, it is easy to immediately correct any erroneous letters via backspacing. An equivalent approach using speech would introduce speech endpointing and recognition delays for each letter in a word. We anticipated this would be too slow and would instead require spelling an entire word at one time. This spelling recognition could introduce further recognition errors into the correction process. An additional technical challenge was that our chosen recognizer, IBM Watson, does not support recognition conditioned on surrounding text context like our keyboard decoder does. This makes correct recognition of uncommon spelled words even more difficult. For these reasons, we decided to support difficult word input using just the midair keyboard. The feasibility of spoken corrections we will explore in offline experiments on the most difficult to correct words we observed in our user study.

3.3 Avoiding Errors and Accelerating Input

Given the lack of tactile keyboard feedback and inaccuracies introduced by hand tracking, keyboard visualization, and hand rendering, we anticipated typing would be more error-prone and slower than on a touchscreen. To help users avoid auto-correction errors and accelerate their input for longer words, we added four prediction slots above the keyboard (Figure 1 left). The predictions slots are 6.5×1.5 cm. We limited the keyboard to four slots in order to keep the buttons above the keyboard reasonably large. In past work [42], increasing the number of slots to five or six only provided small improvements in the character rate and keystroke savings achievable in offline experiments.

The leftmost slot displays the literal text (i.e. exactly the characters typed). Probabilistic input methods can sometimes change a correctly entered word into something else. Various approaches have explored how to take control back from auto-correction. Touch force was used by Weir et al. [51] and Arif and Stuerzlinger [2] to indicate when not to auto-correct. VISAR [10] had a precise selection mode based on dwelling on keys. VelociWatch [42] allowed letters to be locked by long pressing. As is common on phone keyboards, WatchWriter [13] and VelociWatch [42] provided a literal prediction slot based on the exact keys typed. We opted for a similar literal slot approach as it worked well in previous studies [13,42] and may be familiar to users as the feature is common on touchscreen phone keyboards.

The next three slots display the most likely word hypotheses. These hypotheses are either a *prefix completion* or a *recognition alternative*. A prefix completion assumes the user has only typed the start of a word. Since this prefix input may be noisy, we used VelociTap to search for the most probable prefixes under its touch model and language models. Each proposed prefix completion is conditioned on text surrounding the insertion location. For example, if a user typed "rg" at the start of a sentence, the most probable prefix completions might be "the" since "th" is a prefix adjacent to "rg". If however the previous word is "mass", it might propose a word such as "effect" since "mass effect" was common in the language model training data.

A recognition alternative assumes the current tap sequence constitutes an entire word. While the decoder supports character insertions and deletions, typically word alternatives are the same length as the tap sequence. So in the previous example, the decoder might propose a two-letter word such as "eg". Similar to [13,42], our likely slots are populated with the prefix completion or word alternative hypotheses that have the highest log probability. These hypotheses appear in the second, third, and fourth slot in decreasing order. A word is completed by tapping a slot or hitting the space key. The space key selects the text in the second slot (i.e. the most likely word). The text in the second slot is highlighted in yellow. This is similar to how the iOS 12 touchscreen keyboard highlights the correction that will result from tapping space.

3.4 Text Entry Using Speech

When privacy is not an issue, speech recognition has the potential to be a very fast input method. But to be fast as well as natural, speech interfaces typically encourage users to speak a single utterance with their entire phrase or sentence. Given our goal of supporting difficult text input, we anticipated some amount of error correction would be necessary in the recognized block of text. Similar to our keyboard decoder, we would like the speech recognizer to provide likely alternatives for different words in a user's utterance. This led us to choose the IBM Watson service for use in our system. IBM Watson can return a word confusion network that provides a compact and probabilistic representation of the speech recognizer's search. Each cluster in the confusion network contains a list of possible words and their probabilities for each part of the recognition result.

We focused on a design optimized for small amounts of text. We think this is the most pressing need in many VR/AR applications where generating and editing text is not the primary activity. While the recognizer could certainly handle larger utterances (e.g. paragraphs), displaying and correcting within a large amount of text via only hand gestures would be challenging given current limitations in HMD field of view and hand tracking accuracy. While speech-only correction is an option, doing this well requires fine-grained control of the speech decoder (e.g. to support inferring a correction's location [43] and text [45]).

In the case of speech input, the user first presses a record button visible in the virtual environment. This turns on the microphone and starts streaming audio to the speech recognition server. We streamed audio via our university's high speed network to IBM's web service. Streaming audio allowed us to reduce the time users had to wait for recognition results. Once a user stops speaking and word confusion results are available, a stop button appears. Pressing this button displays the recognition result and correction interface.

3.5 Intuitive Speech Error Correction

When a speech recognizer gets a word wrong, often one of its top competing word hypotheses is actually correct. Recognizers also occasionally insert words due to things such as false starts or filler words. Past work has shown word error rate can be reduced by nearly 50% by offering the three best word alternatives plus the ability to delete a given word [44]. We wanted our design to communicate these options to users such that they would not need to learn special correction gestures or take explicit action to see possible correction options. Our design (Figure 2) displays the confusion network similar to Parakeet [44], but uses a smaller number of words (4–6) to allow more accurate midair selection.

Our correction interface consists of a column for each word in the recognition result. The most probable recognition results appear in the top row. Below each word in the best result are up to three other likely alternatives for each word. Each word is a button that can be tapped. Before and after each word in the top row is a small black space button with no label. This button allows a new word to be inserted between existing words. The bottom row contains buttons labeled with an "X" that deletes the word in that column.

To select a different word in a column, a user taps the desired word. This swaps the top word with the tapped word. This does not affect the other alternative words in that column. Our swapping approach allows our design to conserve virtual screen real estate by removing the redundant presentation of the top word present in the Parakeet interface [44] while still preserving the ability to undo erroneous taps on column targets. A slider below the keyboard allows users to adjust

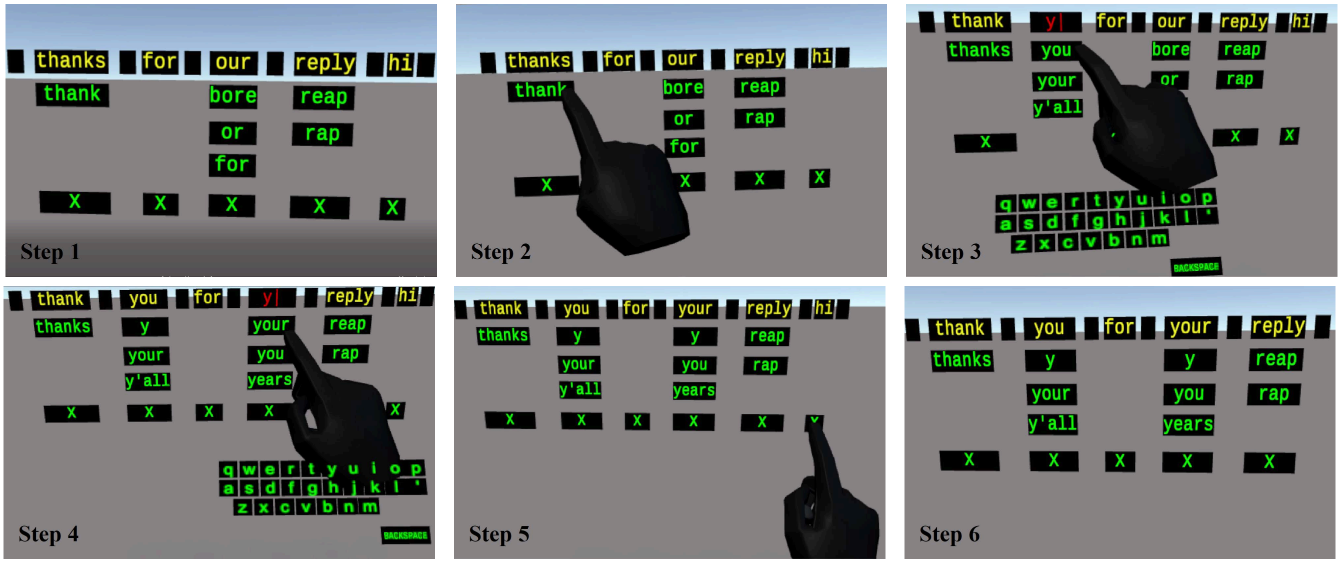


Fig. 2. Example of correcting a sentence using the speech interface. In step 1, the user has spoken “thank you for your reply”. The interface displays the most likely recognition result “thanks for our reply hi” in the top row. In step 2, the user selects the word alternative “thank” and it gets swapped with “thanks”. Next, the user needs to insert between the words “thank” and “for”. In step 3, the user taps on the blank space button appearing between these two words. The keyboard pops up and the user types the letter “y”. The user then selects the prefix completion “you”. In step 4, the user taps the word “our” in the top row. This opens the keyboard and after typing the letter “y”, the user selects the prefix completion “your”. In step 5, the user taps the button labeled X to delete the undesired word “hi”. Step 6 shows the text after all the corrections.

the column centered in the VR display. This slider allows users to scroll through a longer sentence.

If a column does not contain the desired word, a user taps on the top word in the column. We shift the correction interface to center on this column in the virtual space and display the keyboard below the column. During keyboard input, the buttons in a column become the word prediction slots. The top button displays the literal slot which is also the current pending tap sequence. The top button is highlighted red. As the user types, other likely hypotheses appear in the other three buttons in that column. During this time, only interaction with the current column is allowed. A user must tap a slot on the current column to end entry. The tapped slot then becomes the word in the top row. If the user taps on the space between columns to insert a word, an analogous process happens. To preserve consistency with how column words are always tapped to swap them to the top choice in the column, we do not provide a space key. Rather the user concludes typing a word by selecting their desired prediction above the keyboard.

Our design of the keyboard fallback interface is embedded within the display of the confusion network. We felt this design was more consistent and less visually jarring compared to switching to a completely different keyboard screen as in Parakeet. A further difference compared to Parakeet is that our speech correction interface has a probabilistic keyboard that uses the uncertain keyboard tap locations and leverages the surrounding text context via long-span language models.

In both the speech and keyboard-only interfaces, a done button is located about 10 cm below and to the right of the keyboard. Tapping this button marks the completion of a sentence. The done button allows us to include in our entry rate calculations the time users spent reviewing even completely correct speech recognition results, or time spent reviewing the last typed word or correction.

4 User Study

Given a modern speech recognizer, it is obvious that speech input is faster for text requiring few or no corrections, but it is not obvious for harder text that often needs some amount of correction. The primary goal of our user study was to find out if our speech interface design could improve performance on challenging text compared to a competitive non-speech auto-correcting keyboard.

A secondary goal was to add to the limited work on controller-less AR/VR text input using an auto-correcting virtual keyboard with word predictions. To our knowledge, only VISAR [10], Xu et al. [52], and Dudley et al. [9] have explored this. VISAR and Xu et al. did so in AR while we investigate input in VR. VISAR relied on coarse tracking of a user’s wrist, and Xu et al. required users to perform a gesture such as opening and closing the palm to select a key. Dudley et al. used a simulated auto-correction based on knowledge of the stimulus. None of these approaches handled out-of-vocabulary words like our system.

4.1 Participants and Procedure

We recruited 18 participants (11 male, 7 female) via convenience sampling. No participant had uncorrected vision or motor impairments. Participants were aged 18–70 (mean 24.4, sd 14.0), and 16 were right-handed. 14 participants had used VR. All participants were native English speakers. Participants were paid $15. The study took place in a quiet office. We used an HTC Vive HMD with a Leap Motion controller mounted on the front. Audio was recorded from the Vive’s built-in microphone.

The study was a within-subject design with two conditions: Speech and NoSpeech. In Speech, participants first spoke a sentence, then used the word confusion network interface and the virtual keyboard to perform any correction. In NoSpeech, participants typed the entire sentence using the keyboard. Before each condition, we explained to participants how both interfaces worked. Similar to VISAR [10], in both conditions we instructed participants to use the index finger of their dominant hand. Participants first completed a questionnaire asking demographic questions and about their experience with text entry and VR. We seated participants at a desk and helped them adjust the HMD.

Participants entered phrases from the TwitterIV and TwitterOOV phrase sets [42]. All words in TwitterIV phrases are in the 100 K vocabulary used by the keyboard decoder. TwitterOOV phrases have one word that is out-of-vocabulary (OOV). As shown in [42], phrases in TwitterIV are easier for a language model to predict. In contrast, phrases in TwitterOOV are harder to predict. Similar to [42], we used phrases with six or fewer words that did not contain acronyms. Phrases were picked at random. No participant saw the same phrase twice. Participants first practiced on two in-vocabulary and two OOV phrases. They then wrote six in-vocabulary and six OOV phrases. The order of in-vocabulary and OOV phrases were randomized and the conditions

Condition	Entry rate (wpm)	Error rate (CER%)	Borg CR10
NoSpeech	11.1± 2.1 [7.2, 14.5]	1.2± 1.3 [0.0, 4.1]	3.17 ± 1.62 [1.00, 7.00]
Speech	27.9 ± 5.8 [18.7, 37.2]	0.5± 1.0 [0.0, 3.4]	2.11 ± 1.02 [1.00, 5.00]
	t(17) = -13.79, r = 0.96, p < 0.01	t(17) = 2.13, r = 0.46, p < 0.05	$\chi^2(1) = 9, p < 0.01$

Table 1. Results from our study. Results formatted as: mean ± SD[min, max]. The bottom row shows statistical test details.

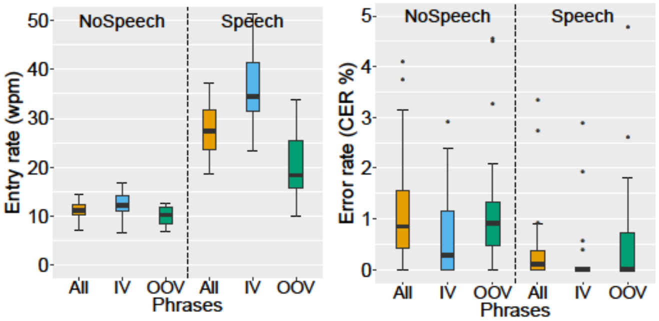


Fig. 3. Entry and error rates in our study. Results over all phrases, in-vocabulary (IV) phrases, and out-of-vocabulary (OOV) phrases.

Metric	All	IV	OOV
Speaking rate (wpm)	151.5	156.4	146.5
Entry rate (wpm)	84.8	81.7	87.9
Error rate (CER%)	10.2	4.4	16.1
Error rate (WER%)	20.8	9.3	32.6

Table 2. Results prior to any correction in the Speech condition. Results for all phrases, in-vocabulary (IV) phrases, and out-of-vocabulary (OOV) phrases.

were counterbalanced. We did not analyze the practice phrases. The study took approximately an hour.

4.2 Results

Overall performance. Table 1 provides numeric results and statistical tests. Figure 3 shows our main entry and error rate results. We calculated *entry rate* in words-per-minute (wpm). We considered a word to be five characters including space. In NoSpeech, entry time was from a participant's first tap until they hit the done button. In Speech, entry time was from a participant's first phoneme until they hit the done button. The entry rate in Speech was faster at 27.9 wpm versus NoSpeech at 11.1 wpm. This difference was significant (Table 1).

The difficulty of the target text did seem to slow participants down in both conditions. In NoSpeech, the entry rate was 12.0 wpm for in-vocabulary phrases versus 10.1 wpm for OOV phrases (Figure 3 left). This difference was significant ($t(17) = 4.90, r = 0.77, p < 0.001$). In Speech, the entry rate was 35.7 wpm for in-vocabulary phrases versus 19.8 wpm for OOV phrases (Figure 3 left). This difference was also significant ($t(17) = 7.72, r = 0.88, p < 0.001$).

We measured *error rate* using Character Error Rate (CER). CER is the number of character insertions, deletions, and substitutions needed to change a participant's final text into the reference text divided by the characters in the reference. We measured the CER of a participant's final text after completing any corrections. Participants were less accurate in NoSpeech with a CER of 1.2% versus 0.5% in Speech. This difference was significant (Table 1). In Speech, error rate was 0.3% for in-vocabulary phrases versus 0.7% for OOV phrases (Figure 3 right). This difference was significant ($t(17) = -2.11, r = 0.46, p < 0.05$). In NoSpeech, error rate was 1.2% for in-vocabulary phrases versus 1.3% for OOV phrases. This difference was not significant.

Participants rated their exertion on the Borg CR10 scale [4]. The mean rating was 3.17 in NoSpeech and 2.11 in Speech. This difference was significant ($\chi^2(1)$=9.0, p < 0.01). Thus it appears the speech interface did help reduce the "gorilla arm" [5] problem that is common in midair interactive systems.

Taken together, our entry and error rates show that our speech interface was successful at providing better performance on challenging text compared to our keyboard-only interface. The speech interface was 2.5 times faster than the keyboard-only interface while maintaining a similar and low corrected error rate. Further, participants reported less fatigue using the speech interface.

As might be expected, the in-vocabulary phrases were much faster to enter than OOV phrases in Speech at 1.8 times faster. This speed differential was more muted in NoSpeech at 1.2 times faster. This suggests participants successfully avoided some of the overheads of locating or fixing errors in the keyboard-only interface by careful typing and use of the literal slot. Providing similar ways to preemptively avoiding recognition errors would likely improve the speech interface's performance on OOV phrases. This might be possible by having users both speak and spell words they think may be a problem when first speaking their sentence. However such a feature would require changes to the underlying speech recognition algorithm.

Detailed performance analysis. Using the speech interface, participants' average speaking rate (not including recognition latency or correction) was 151.5 wpm (Table 2). Our system streamed audio to the IBM Watson web service. After participants stopped speaking, they had to wait for the recognizer to detect the silence at the end of their speech, recognize their speech, and return the word confusion network. This introduced an average delay of 1.8 s (measured from a participant's last phoneme until the result arrived at our system). Assuming participants did not need to correct or even confirm results, this would result in a best-case entry rate of 84.8 wpm. Our actual entry rate, including any corrections and a final confirmation step was 27.9 wpm (Table 1). This shows both the importance of low latency speech recognition, but also the substantial time costs associated with error correction.

Table 2 shows that the CER of speech was much higher at 10.2% before any correction versus 0.5% after correction. The out-of-vocabulary phrases exhibited speech recognition character error rates of about four times that of the in-vocabulary phrases. We also computed the Word Error Rate (WER). WER is analogous to CER but on the word level. We see that in OOV phrases nearly one in three words required correction. Overall, we found that OOV phrases were successful at creating error rates that substantially increased the use of correction features.

Figure 4 visualizes participants' entry and error rate before and after correction and confirmation. We see that participants experienced quite variable speech recognition accuracy with error rates ranging from 4% to 25%. All participants were able to substantially reduce errors made by the speech recognizer to below 5%. Half of the participants were able to achieve a 0% CER.

46% of phrases were recognized with no errors. The average entry rate of these phrases at the time when the speech recognition result was first available was 81.3 wpm. However, also including the time it took participants to hit the done button resulted in a drop to 42.0 wpm. Thus, participants were spending considerable time either reviewing the result or executing a tap on the done button.

Correction feature usage. We analyzed what correction features participants made use of. In NoSpeech, participants selected the 1-likely slot (either by tapping the second prediction slot or by hitting space) 46.4% of the time, 2-likely slot 32.9% of the time, 3-likely slot 20.1% of the time, and literal slot only 0.6% of the time. In Speech, participants selected the literal slot 33.3% of the time, 1-likely slot 50.0% of the time, 2-likely slot 14.0% of the time, and 3-likely slot 2.6% of the time. We suspect the literal slot was used more frequently in Speech because the keyboard was mostly used for correcting errors.

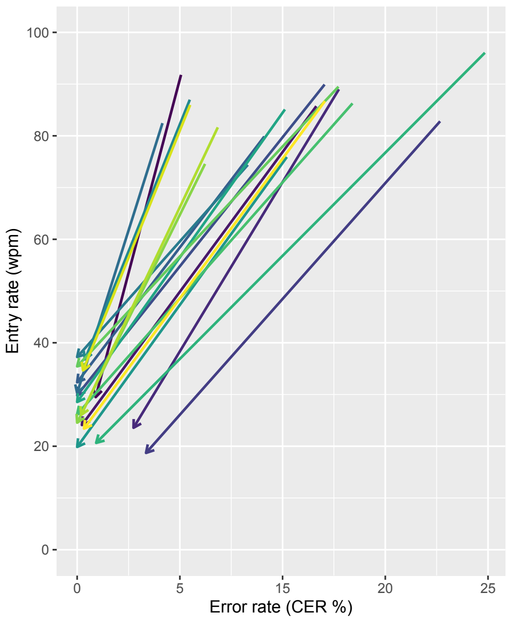


Fig. 4. Entry rate and error rate of each participant in the Speech condition. The right end of each line is before correction of the result while the left end is after making corrections and confirming the result.

	NoSpeech
+	"The suggested words helped a ton with speed"
+	"Auto-suggestions were very intelligent and helped considerably"
+	"I could type whatever I wanted"
+	"Having a large keyboard with keys large enough for my virtual fingers"
+	"The elevated keyboard made it easy to see my keys and text at the same time"
–	"Sometimes my hand would press a key next to the key I want"
–	"The system struggled to determine which finger I was using"
–	"Arm got tired fairly quickly"
–	"Hands were the same color as the keyboard. It was difficult to keep track at times."
–	"I hated the autocorrect changing what I had typed"

	Speech
+	"Easier and quicker than typing everything"
+	"Speech seemed to be a lot faster"
+	"It was easy to understand and convenient to use"
+	"The very clear visuals and knowing where to press."
+	"Arm didn't tire. Also helped me make sure I was talking clearly plus enunciating"
–	"Editing did not allow using a partial word"
–	"Uncommon words were tough to pronounce, and didn't came up through speech"
–	"I didn't like how I had to have my finger/hand in a certain way to type."
–	"It felt you needed to put your finger through the key"
–	"When I made a mistake, or it did, correcting the text was slow and frustrating"

Table 3. Some positive and negative comments about each interface.

These errors involved words that were hard for the speech recognizer and were thus unlikely to be predicted by the keyboard recognizer.

Recall that immediately after recognizing a word, our keyboard displayed a delete-word button in NoSpeech. Across the whole study, this button was only used six times. Instead, participants often repeatedly hit the backspace key to erase an erroneous word. In NoSpeech, participants used the backspace key one or more times in 63.2% of OOV phrases compared to 36.8% for in-vocabulary phrases. In the Speech condition, participants used backspace in 95.6% of OOV phrases compared to 4.4% for in-vocabulary phrases. This shows that for more difficult text, participants focused on exactly typing their desired letters rather than relying on auto-correction or word predictions.

We wondered how often participants entered a sentence in the Speech condition using only the word confusion network interface (i.e. without using the keyboard). 115 out of 214 phrases had one or more speech recognition errors. Participants corrected 30.4% of these phrases using only the word confusion network. For the remaining 70% phrases, either the answer was not in the word confusion network or users deleted a partially correct answer and retyped. This suggests to further improve performance, we may need to focus on ways to speed the entry of words not in the recognizer's top results.

The speech interface had an "X" button for deleting a cluster and a blank button for inserting a word. The delete button was used one or more times in 32.2% of phrases. This is mainly because the speech recognition results sometimes split a word spoken by a participant into several phonetically similar words (e.g. *your* was recognized as *you are*). The insert button was only used in 1.9% of phrases.

Recall we have a slider in the speech correction interface. Only one participant used the slider. This was because our input phrases were relatively short (4 – 6 words) and thus the WCN almost always appeared within the participants' field of view with the leftmost and rightmost columns reachable by a user's virtual hands.

Subjective feedback. After each condition, participants rated a series of statements on a 5-point Likert scale (1 = strongly disagree, 5 = strongly agree). We tested for significance using Friedman's test. The mean rating for the statement "I thought I entered text *quickly*" was 3.56 in NoSpeech and 3.83 in Speech. This difference was not significant ($\chi^2(1)$=0.0, p=1.0). The mean rating for the statement "I thought I entered text *accurately*" was 4.0 in NoSpeech and 3.89 in Speech. This also was not significant ($\chi^2(1)$=0.11, p=0.73). The mean rating for the statement "The interface provided accurate visual feedback of my hand(s)" was 3.83 in NoSpeech and 3.67 in Speech. The difference was not statistically significant (Friedman's test, $\chi^2(1) = 0.4$, $p = 0.53$). The mean rating for the statement "The interface detected key press accurately" was 3.5 in NoSpeech and 3.78 in Speech. The difference was not statistically significant (Friedman's test, $\chi^2(1) = 2$, $p = 0.13$). Taken together, the Likert results show slightly positive sentiment about the various aspects of both interfaces. But it seems participants saw definite room for improvement for both interfaces.

After each condition, we asked participants to point out a positive or negative aspect of the interface. Table 3 shows a representative sample of these comments. The most common positive comments were about the utility of the predictions, accuracy of auto-correct, and the speed of speech input. The most common negative comments were on the difficulty of accurately tapping midair targets in both conditions, and the low speech recognition accuracy on uncommon words.

At the end of the study, participants specified their preferred condition in terms of quickness, accuracy, minimal effort, and overall. In general, participants preferred Speech:

- Quickness — Speech 14, NoSpeech 4
- Accuracy — Speech 10, NoSpeech 8
- Effort — Speech 17, NoSpeech 1
- Overall — Speech 15, NoSpeech 3

5 Offline Experiment on Spoken Corrections

In our study, we relied exclusively on a midair virtual keyboard to correct speech recognition errors. But as previously discussed, a possible alternative is to have users speak or spell the misrecognized word. We conducted an exploratory study to investigate the efficacy of using spoken or spelled words to correct errors. As we will see, speech recognition errors were common on spoken and spelled words. Thus we further explored if we can re-rank the speech recognition hypotheses using a neural language model.

We focused on the most difficult cases where our word confusion network correction interface failed. We identified 57 words from sentences in our user study for which users encountered speech recognition errors and were forced to resort to keyboard correction because the reference word was not in the correction interface. These words were mostly words that were difficult to pronounce and probably also not in the vocabulary of the IBM Watson recognizer. Some examples include: *eunjung*, *swaraj's*, *auchinleck*, *bourre*, *tabinof*, and *chopt*.

5.1 Data Collection

We highlighted the problematic words in an audio collector application that showed each word in the context of its reference sentence. We instructed people to speak each of the 57 words in three distinct ways:

- SPEAK: Speak the word normally, e.g. by speaking "tabinof".
- SPELL-QUICKLY: Spell each letter of the word quickly, e.g. by speaking "T-A-B-I-N-O-F".
- SPELL-CAREFULLY: Similar to the previous condition except users were told to enunciate each letter carefully, as if someone was listening and trying to write down the letter sequence.

We recorded audio from five users including the two authors of this paper. Users recorded their audio using their own computer and headset microphone. We used the IBM Watson speech-to-text service to transcribe the recordings. We used the same parameters for Watson as in our main study. We logged the transcription results including the best and all alternative hypotheses (i.e. the confusion network returned by Watson). Each hypothesis was a word or character along with its probability. Note that the IBM Watson service does not provide a way to specify surrounding text context for a recognition request. It also does not provide an explicit mode for recognizing spelling. This makes recognition harder than it might be with a speech engine that supports conditioning its language model on surrounding text or that has a spelling-specific model. Nonetheless, these limitations are the same between all audio conditions and still allows us to measure the relative accuracy of the three speaking methods.

5.2 Ranking Hypotheses

For the two spelling conditions, we took all the character alternatives in each position of the word confusion network and generated all possible words from the character alternatives. We scored each word by multiplying the probabilities given in the confusion network for its constituent characters. For the SPEAK condition, we simply considered all the alternative words suggested by the speech recognizer in the single cluster of the word confusion network.

We ranked the words by their scores and calculated the word error rate (WER). Given we are recognizing a single word, WER is simply the percentage of corrections that were recognized incorrectly. We computed the WER of the top choice (the 1-best) and the 4-best oracle WER. The latter represents how often you would get a word wrong in a user interface providing the best and next three best hypotheses.

5.3 Hypothesis Re-ranking

We were interested if we could improve accuracy by re-ranking all possible word alternatives by applying a language model to take into account the surrounding text in the sentence. For this we used the bidirectional neural language model BERT [7]. BERT has proven useful in a range of natural language processing tasks where an entire sentence is available at inference time (e.g. classifying a tweet's sentiment).

BERT is trained in part by taking sentences and masking individual words. We make use of this to generate a BERT probability for any given sentence. This was done by masking each word in a sentence one-at-a-time and finding the log probability of the masked word under the BERT model. Adding the log probabilities of all the words yields a sentence's log probability. We used the pre-trained `bert-base-uncased`[1] model with 12 transformers for our purposes.

To re-rank a list of hypotheses for a spoken correction, we took the reference text and replaced the reference word at the correction location with each possible correction hypothesis. We then added each hypothesis' BERT sentence log probability to its original log probability based on IBM Watson's confusion network. Note that the contribution of BERT to the speech recognition probability could be varied by introducing a scale factor (i.e. multiplying the BERT probability by a configurable parameter). We did not investigate that here and instead simply used a scale factor of one. By adding BERT's log probability to Watson's log probability we obtain a new ordering of the hypotheses that takes into account how the correction fits within the entire sentence.

	Word Error Rate (WER%)			
	1-best	1-best	4-best	4-best
Condition		+ BERT		+ BERT
SPELL-CAREFULLY	62.5	51.9	48.4	43.5
SPELL-QUICKLY	77.2	72.3	69.5	68.8
SPEAK	74.4	70.5	65.3	62.1

Table 4. 1-best and 4-best word error rates calculated from a list of all possible words constructed from Watson's hypothesis space for spoken corrections. We also re-ranked the words using BERT.

5.4 Results

Table 4 shows the main results. The SPELL-CAREFULLY condition was the most accurate. When users spelled carefully, the 1-best WER was 62.5% and the 4-best WER was 48.4%. Re-ranking the word alternatives with BERT reduced the 1-best WER to 51.9% and the 4-best WER to 43.5%. This shows the advantage of leveraging surrounding text context via a powerful bidirectional neural language model.

In SPELL-QUICKLY, the 1-best WER was 77.2% and the 4-best WER was 69.5%. Similar to the previous condition, re-ranking with BERT reduced the 1-best WER to 72.3 and the 4-best WER to 68.8%. The error rate substantially increased for quick spellings. This suggests if spelling correction is to be added to the interface, users need to be encouraged to clearly pronounce each letter.

In SPEAK, the 1-best WER was 74.4% and 4-best WER was 65.3%. Similar to the other conditions, using BERT reduced the 1-best WER to 70.5% and the 4-best WER to 62.1%. Just speaking the word performed similar to spelling it quickly, but not as well as spelling it carefully.

We calculated how fast users spoke correction by using the time of the first and last phoneme provided by IBM Watson. It took users about 4.0 s to speak a correction in SPELL-CAREFULLY, 1.8 s in SPELL-QUICKLY, and 0.7 s in SPEAK. We measured the recognition latency on the correction by measuring the time between sending the request to receiving the word confusion network result. The latency was 3.9 s in SPELL-CAREFULLY, 2.6 s in SPELL-QUICKLY, and 2.0 s in SPEAK.

Overall, even after re-ranking the carefully spelled hypotheses with BERT and assuming the interface provided the top four options, 43.5% of correction attempts would still have failed. While spelling carefully was the slowest spoken correction technique, it would still be much quicker than typing most words on a midair keyboard. However, this speed advantage could disappear if repeated spellings are required. This result suggests caution in solely relying on spelling as a correction method for difficult words. Perhaps having the user provide additional acoustic information could help the recognizer (e.g. military phonetic spelling). It may also help if the speech recognizer made use of the surrounding text context or supported a dedicated spelling model.

6 DISCUSSION

Given the limited research on text entry using speech in virtual environments, we think our findings provide useful insights in how to design text entry for virtual environments. In particular, we explored how to enable efficient text entry in virtual environments without the use of auxiliary input devices. Our system relied only on midair gestures and speech input. Further, we aimed to design a system that could be used with little or no training.

Overall we found that speaking a sentence and then correcting errors using a midair interface was much faster at 28 wpm compared to 11 wpm for typing the whole sentence. Even better, we were able to support this speed on text that often contained out-of-vocabulary (OOV) words. For the subset of phrases without OOVs, the entry rate was faster at 36 wpm. This is competitive with other speech systems such as SWIFTER [34] (23.6 wpm) and SpeeG2 [18] (21 wpm).

However, even on in-vocabulary phrases, our entry rate was substantially slower than the 153 wpm reported by Ruan et al. on a touchscreen phone [35]. There are a number of possible reasons for this. First, we relied on midair tapping which is less accurate and higher latency than tapping on a touchscreen. Second, in our study, only 46% of sentences

[1] https://github.com/google-research/bert

were perfectly recognized. While Ruan et al. does not report how many sentences were perfectly recognized, they do report the entry rate was 179 wpm prior to any correction. Since this rate only reduced to 153 wpm overall, this suggests most sentences required no correction. It is also not clear whether they included the time participants spent confirming completely correct recognitions (something we included in our entry rate). Third, we incurred several seconds of latency waiting for IBM Watson to endpoint a users speech and return the confusion network. This delay could be reduced by introducing an explicit "push to talk" action, e.g. using a gesture such as lifting and lowering your hand to turn the mic on and off.

Our speech interface was particularly slow for uncommon words where the correct word did not appear in the likely word alternatives. In these cases, users had to fallback to midair keyboard correction. In our offline experiment, we explored the possibility of using spoken corrections instead for fixing these uncommon words. Our initial offline results show that even employing a powerful neural language model to re-rank IBM Watson's spoken correction hypotheses resulted in a correction method that would often need correction itself. However, this finding needs further validation with more users, ideally tested inside a user interface with a speech engine specifically designed to handle spoken or spelled corrections.

In our study, while we considered input of uncommon OOV words, our phrases only contained 26 letters of the alphabet and apostrophe. Performance may be slower for other types of text containing, for example, passwords, special symbols, or numbers. The input of numbers, and special characters might be important for some tasks, e.g. entering mathematical or chemical formulas. It remains to be seen how well our system would support input of such text.

We had participants enter relatively short phrases (four to six words). While our speech correction interface can handle the input of longer phrases, the entire WCN might not fit in a user's field of view. Our slider can be used to scroll to the left and rightmost elements in the WCN, but it might also be possible to automate scrolling through the WCN at a speed that can be adjusted according to a user's needs.

We think our speech interface will be useful for scenarios such as short messaging and search queries. In particular, search queries may often contain difficult to recognize words. However, we suspect our interface may be too tiring for long-term use. It is also not suited for creating large passages of text. This would require design of appropriate features to allow users to move around, select, and correct text. It might also require continuous recognition and result display rather than the utterance-at-a-time approach we employed.

Lastly, we conducted a single hour session and participants completed only 12 text entry tasks in each condition. While the shorter session and a smaller number of input tasks were sufficient to prove the efficiency of our system for walk-up-and-use scenarios, our study did not reveal how users adapted to the system. One might expect performance to increase in both conditions with practice, showing this would require a longitudinal evaluation.

6.1 Design Implications

We found our correction interface was quite successful at allowing users to correct errors. In particular, 30% of sentences could be corrected without resorting to the keyboard. This is a marked improvement to the 10% reported in [44]. Based on our study, we think our design could be improved in a number of ways:

1. **Reduce column size.** We found the third most likely slot in the speech and keyboard interface was rarely used. In the study participants sometimes accidentally hit an adjacent button. Particularly for the speech interface, it may be better to have bigger or more separated buttons in each column to avoid such errors.

2. **Different hand and keyboard color.** The color of our virtual keyboard and the virtual representation of the users' hands were similar. In particular, one participant in our study had problems with separation. While none of the other participants complained about this issue, we think it would be better to color the virtual representation of hands and the keyboard differently.

3. **Allow modification of words.** We observed that sometimes the speech recognizer returned words that were close to, but not identical to the target word (e.g. "closed" when the user wanted "closing"). In such cases, users had to retype the entire word. One possible design change would be to let users partially backspace a recognized word and type the remaining letters. But whether this approach ends up saving time needs further investigation. Providing a richer set of correction options could in fact slow users down if it creates indecision about the best course of action.

4. **Spoken word corrections.** Another interesting extension would allow users to speak or spell words similar to SWIFTER [34] and SpeeG2 [18]. Even further, the system could support both speaking *and* spelling a word (e.g. "dog D-O-G"). This has been shown to be more accurate than just speaking or just spelling a word [47]. Speaking or spelling words could be used instead of our current keyboard based insertion or substitution of words. Prediction slots would be filled with the most likely recognized words based on a spoken correction. Our offline experiments suggest for difficult words, asking users to spell the word carefully may be the most accurate option, especially when coupled with a powerful bidirectional neural language model.

5. **Improved speech recognizer.** We could train and host our own speech recognizer rather than relying on a commercial recognizer. This could reduce network latency. However, using a commercial recognizer has the advantage that the recognizer may be trained on substantially more data than is available to researchers. Thus a commercial recognizer may be more accurate. But an advantage of an in-house research recognizer is that we could add support for features like speaking and spelling words that may not be available in commercial APIs. It may also allow exploration of avoiding errors in the first place by allowing users to provide spelling of difficult words in their initial utterance.

6. **Speech-aware keyboard decoder**. When correcting speech errors with the keyboard, our keyboard only made use of the noisy tap sequence plus the text to the left and to the right of the location being edited. It does not leverage the probabilistic information in the speech recognition result for the sentence being corrected. For example, while the user's desired word may appear somewhere in the word confusion network result, it may not have a high enough probability to appear in our correction interface. It is possible this information could improve the keyboard's predictions.

7 Conclusion

To date, speech-assisted text input in virtual environments has seen little investigation. While there are many speech-based text input studies on desktop and mobile devices, studies of speech input in VR HMDs are lacking. We presented a text entry system for virtual environments that allows users to type in midair on a familiar QWERTY virtual keyboard. We investigated the speed, accuracy, ergonomics, and user satisfaction of our speech-assisted text entry method. Our work informs the design of text entry interfaces that support the input of even challenging text, including uncommon words such as proper names. Moreover, by augmenting our interface with speech, we created a significantly faster and less exerting input method for virtual reality.

We found even on text with difficult words (which has not previously been investigated in any speech-based VR/AR interfaces), users could combine spoken input and tapping on midair virtual buttons to write at 28 wpm at a low error rate of 0.5%. To our knowledge, we are the first to investigate speech-based text entry in a VR head-mounted display without the use of auxiliary input devices. We believe with the burgeoning consumer demand of head-mounted display devices, our design and study findings will inform future VR and AR interfaces that require efficient text input.

Acknowledgments

This material is based upon work supported by the NSF under Grant No. IIS-1750193.

References

[1] S. Ahn, S. Heo, and G. Lee. Typing on a smartwatch for smart glasses. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, ISS '17, pp. 201–209. ACM, New York, NY, USA, 2017. doi: 10.1145/3132272.3134136

[2] A. S. Arif and W. Stuerzlinger. Pseudo-pressure detection and its use in predictive text entry on touchscreens. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, OzCHI '13, pp. 383–392. ACM, New York, NY, USA, 2013. doi: 10.1145/2541016.2541024

[3] C. Boletsis and S. Kongsvik. Controller-based text-input techniques for virtual reality: An empirical comparison. *International Journal of Virtual Reality*, 19(3):2–15, Oct. 2019. doi: 10.20870/IJVR.2019.19.3.2917

[4] G. Borg. *Borg's Perceived Exertion and Pain Scales.* Human Kinetics, 1998.

[5] S. Boring, M. Jurmu, and A. Butz. Scroll, tilt or move it: Using mobile phones to continuously control pointers on large public displays. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, pp. 161–168, 2009.

[6] D. A. Bowman, C. J. Rhoton, and M. S. Pinho. Text input techniques for immersive virtual environments: An empirical comparison. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 46(26):2154–2158, 2002. doi: 10.1177/154193120204602611

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota, June 2019. doi: 10.18653/v1/N19-1423

[8] T. J. Dube and A. S. Arif. Text entry in virtual reality: A comprehensive review of the literature. In *International Conference on Human-Computer Interaction*, pp. 419–437. Springer, 2019.

[9] J. Dudley, H. Benko, D. Wigdor, and P. O. Kristensson. Performance envelopes of virtual keyboard text input strategies in virtual reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 289–300, 2019. doi: 10.1109/ISMAR.2019.00027

[10] J. J. Dudley, K. Vertanen, and P. O. Kristensson. Fast and precise touch-based text entry for head-mounted augmented reality with variable occlusion. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 25(6), 12 2018. doi: 10.1145/3232163

[11] A. M. Feit, S. Sridhar, C. Theobalt, and A. Oulasvirta. Investigating multi-finger gestures for mid-air text entry. In *ACM womENcourage*, 2015.

[12] M. Foley, G. Casiez, and D. Vogel. Comparing smartphone speech recognition and touchscreen typing for composition and transcription. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1—11. ACM, New York, NY, USA, 2020. doi: 10.1145/3313831.3376861

[13] M. Gordon, T. Ouyang, and S. Zhai. WatchWriter: Tap and gesture typing on a smartwatch miniature keyboard with statistical decoding. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, CHI '16, pp. 3817–3821. ACM, New York, NY, USA, 2016. doi: 10.1145/2858036.2858242

[14] J. Grubert, L. Witzani, E. Ofek, M. Pahud, M. Kranz, and P. O. Kristensson. Effects of hand representations for typing in virtual reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 151–158. IEEE, 2018. doi: 10.1109/VR.2018.8446250

[15] J. Grubert, L. Witzani, E. Ofek, M. Pahud, M. Kranz, and P. O. Kristensson. Text entry in immersive head-mounted display-based virtual reality using standard keyboards. *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 159–166, 2018. doi: 10.1109/VR.2018.8446059

[16] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. doi: 10.1109/MSP.2012.2205597

[17] L. Hoste, B. Dumas, and B. Signer. SpeeG: A multimodal speech- and gesture-based text input solution. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pp. 156––163. ACM, New York, NY, USA, 2012. doi: 10.1145/2254556.2254585

[18] L. Hoste and B. Signer. SpeeG2: A speech- and gesture-based interface for efficient controller-free text input. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ICMI '13, pp. 213—220. ACM, New York, NY, USA, 2013. doi: 10.1145/2522848.2522861

[19] H. Jiang and D. Weng. HiPad: Text entry for head-mounted displays using circular touchpad. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 692–703, 03 2020. doi: 10.1109/VR46266.2020.00092

[20] C.-M. Karat, C. Halverson, D. Horn, and J. Karat. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pp. 568–575. ACM, New York, NY, USA, 1999. doi: 10.1145/302979.303160

[21] P. Knierim, V. Schwind, A. M. Feit, F. Nieuwenhuizen, and N. Henze. 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, pp. 1—9. ACM, New York, NY, USA, 2018. doi: 10.1145/3173574.3173919

[22] K. Kurihara, M. Goto, J. Ogata, and T. Igarashi. Speech pen: Predictive handwriting based on ambient multimodal recognition. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, CHI '06, pp. 851–860. ACM, New York, NY, USA, 2006. doi: 10.1145/1124772.1124897

[23] K. Larson and D. Mowatt. Speech error correction: The story of the alternates list. *International Journal of Speech Technology*, 6:183–194, 01 2003. doi: 10.1023/A:1022342732234

[24] M. Lee and W. Woo. ARKB: 3D vision-based augmented reality keyboard. In *Proceedings of the 13th International Conference on Artificial Reality and Telexistence*, ICAT'03, 2003.

[25] I. S. MacKenzie and R. W. Soukoreff. Phrase sets for evaluating text entry techniques. In *Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pp. 754–755. ACM, New York, NY, USA, 2003. doi: 10.1145/765891.765971

[26] L. Mangu, E. Brill, and A. Stolcke. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14(4):373–400, 2000.

[27] A. Markussen, M. R. Jakobsen, and K. Hornbæk. Selection-based mid-air text entry on large displays. In *IFIP Conference on Human-Computer Interaction – INTERACT 2013*, pp. 401–418. Springer, Berlin, Heidelberg, 2013.

[28] A. Markussen, M. R. Jakobsen, and K. Hornbæk. Vulture: A mid-air word-gesture keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pp. 1073—1082. ACM, New York, NY, USA, 2014. doi: 10.1145/2556288.2556964

[29] M. McGill, D. Boland, R. Murray-Smith, and S. Brewster. A dose of reality: Overcoming usability challenges in VR head-mounted displays. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 2143–2152. ACM, New York, NY, USA, 2015. doi: 10.1145/2702123.2702382

[30] J. Ogata and M. Goto. Speech repair: Quick error correction just by using selection operation for speech input interfaces. In *Proceedings of the International Conference on Spoken Language Processing*, pp. 133–136, September 2005.

[31] A. Otte, T. Menzner, T. Gesslein, P. Gagel, D. Schneider, and J. Grubert. Towards utilizing touch-sensitive physical keyboards for text entry in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1729–1732, 2019. doi: 10.1109/VR.2019.8797740

[32] S. Oviatt. Taming recognition errors with a multimodal interface. *Communications of the ACM*, 43(9):45–51, 2000.

[33] D.-M. Pham and W. Stuerzlinger. HawKEY: Efficient and versatile text entry for virtual reality. In *25th ACM Symposium on Virtual Reality Software and Technology*, VRST '19. ACM, New York, NY, USA, 2019. doi: 10.1145/3359996.3364265

[34] S. Pick, A. S. Puika, and T. W. Kuhlen. SWIFTER: Design and evaluation of a speech-based text input metaphor for immersive virtual environments. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 109–112. IEEE, 2016.

[35] S. Ruan, J. O. Wobbrock, K. Liou, A. Ng, and J. A. Landay. Comparing speech and keyboard text entry for short messages in two languages on touchscreen phones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):159:1–159:23, Jan. 2018. doi: 10.1145/3161187

[36] D. Schneider, A. Otte, T. Gesslein, P. Gagel, B. Kuth, M. S. Damlakhi, O. Dietz, E. Ofek, M. Pahud, P. O. Kristensson, J. Müller, and J. Grubert. ReconViguRation: Reconfiguring physical keyboards in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 25(11):3190–3201, 2019. doi: 10.1109/TVCG.2019.2932239

[37] R. W. Soukoreff and I. S. MacKenzie. Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pp. 319—320. ACM, New York, NY, USA, 2001. doi: 10.1145/634067.634256

[38] M. Speicher, A. M. Feit, P. Ziegler, and A. Krüger. Selection-based text entry in virtual reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 1—–13. ACM, New York, NY, USA, 2018. doi: 10.1145/3173574.3174221

[39] S. Sridhar, A. M. Feit, C. Theobalt, and A. Oulasvirta. Investigating the dexterity of multi-finger input for mid-air text entry. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 3643—3652. ACM, New York, NY, USA, 2015. doi: 10.1145/2702123.2702136

[40] B. Suhm, B. Myers, and A. Waibel. Multimodal error correction for speech user interfaces. *ACM Transactions on Computer-Human Interaction*, 8(1):60–98, Mar. 2001. doi: 10.1145/371127.371166

[41] K. Vertanen, C. Fletcher, D. Gaines, J. Gould, and P. O. Kristensson. The impact of word, multiple word, and sentence input on virtual keyboard decoding performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 626:1–626:12. ACM, New York, NY, USA, 2018. doi: 10.1145/3173574.3174200

[42] K. Vertanen, D. Gaines, C. Fletcher, A. M. Stanage, R. Watling, and P. O. Kristensson. VelociWatch: Designing and evaluating a virtual keyboard for the input of challenging text. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pp. 591:1–591:14. ACM, New York, NY, USA, 2019. doi: 10.1145/3290605.3300821

[43] K. Vertanen and P. O. Kristensson. Automatic selection of recognition errors by respeaking the intended text. In *ASRU '09: IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 130–135, December 2009. doi: 10.1109/ASRU.2009.5373347

[44] K. Vertanen and P. O. Kristensson. Parakeet: A continuous speech recognition system for mobile touch-screen devices. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pp. 237–246. ACM, New York, NY, USA, 2009. doi: 10.1145/1502650.1502685

[45] K. Vertanen and P. O. Kristensson. Getting it right the second time: Recognition of spoken corrections. In *Proceedings of the 3rd IEEE Workshop on Spoken Language Technology*, SLT'10, pp. 277–282, December 2010.

[46] K. Vertanen and P. O. Kristensson. A versatile dataset for text entry evaluations based on genuine mobile emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices & Services*, MobileHCI '11, pp. 295–298. ACM, New York, NY, USA, 2011. doi: 10.1145/2037373.2037418

[47] K. Vertanen and P. O. Kristensson. Spelling as a complementary strategy for speech recognition. In *Proceedings of the International Conference on Spoken Language Processing*, INTERSPEECH '12, September 2012.

[48] K. Vertanen, H. Memmi, J. Emge, S. Reyal, and P. O. Kristensson. VelociTap: Investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '15, pp. 659–668. ACM, New York, NY, USA, 2015. doi: 10.1145/2702123.2702135

[49] J. Walker, B. Li, K. Vertanen, and S. Kuhl. Efficient typing on a visually occluded physical keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 5457–5461. ACM, New York, NY, USA, 2017. doi: 10.1145/3025453.3025783

[50] C.-Y. Wang, W.-C. Chu, P.-T. Chiu, M.-C. Hsiu, Y.-H. Chiang, and M. Y. Chen. PalmType: Using palms as keyboards for smart glasses. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pp. 153–160. ACM, New York, NY, USA, 2015. doi: 10.1145/2785830.2785886

[51] D. Weir, H. Pohl, S. Rogers, K. Vertanen, and P. O. Kristensson. Uncertain text entry on mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pp. 2307–2316. ACM, New York, NY, USA, 2014. doi: 10.1145/2556288.2557412

[52] W. Xu, H. Liang, A. He, and Z. Wang. Pointing and selection methods for text entry in augmented reality head mounted displays. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 279–288, 2019. doi: 10.1109/ISMAR.2019.00026

[53] W. Xu, H.-N. Liang, Y. Zhao, T. Zhang, D. Yu, and D. Monteiro. RingText: Dwell-free and hands-free text entry for mobile head-mounted displays using head motions. *IEEE transactions on visualization and computer graphics*, 25(5):1991–2001, 2019.

[54] X. Yi, C. Yu, M. Zhang, S. Gao, K. Sun, and Y. Shi. ATK: Enabling ten-finger freehand typing in air based on 3d hand tracking data. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*, UIST '15, pp. 539–548. ACM, New York, NY, USA, 2015. doi: 10.1145/2807442.2807504

[55] C. Yu, Y. Gu, Z. Yang, X. Yi, H. Luo, and Y. Shi. Tap, dwell or gesture?: Exploring head-based text entry techniques for hmds. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 4479–4488. ACM, New York, NY, USA, 2017. doi: 10.1145/3025453.3025964

[56] C. Yu, K. Sun, M. Zhong, X. Li, P. Zhao, and Y. Shi. One-dimensional handwriting: Inputting letters and words on smart glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 71—82. ACM, New York, NY, USA, 2016. doi: 10.1145/2858036.2858542

[57] S. Zhai and P. O. Kristensson. The word-gesture keyboard: Reimagining keyboard interaction. *Communications of the ACM*, 55(9):91–101, September 2012. doi: 10.1145/2330667.2330689