TapGazer: Text Entry with Finger Tapping and Gaze-directed Word Selection

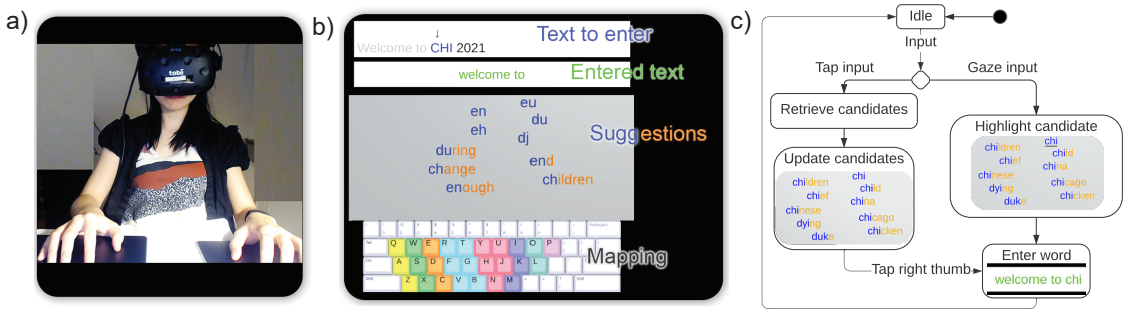

Fig. 1. a) Physical setup: The user enters text without needing to see hands or keyboard, by tapping on a surface and resolving ambiguity between candidate words via gaze selection. b) Visual interface: Fingers are mapped to multiple letters (see colors at bottom); the central area shows candidate words corresponding to the current input sequence of finger taps. Users can select a word by gazing at it and tapping the right thumb. c) State machine of Tapgazer with gaze selection and word completion.

In the years to come, smartphones may be replaced by inexpensive and lightweight eyeglasses capable of displaying augmented reality. Ideally those future glasses will incorporate gaze detection, and will also track hand and finger movements. Wearers of those glasses should then be able to type text simply by tapping their fingers on any surface. In anticipation of this future, we present TapGazer, a text entry system in which users type by tapping their fingers, without needing to look at their hands or be aware of their hand position. Ambiguity is resolved at the word level, by using gaze for word selection. In the absence of gaze tracking, disambiguation can be effected by additional taps. User studies show that TapGazer works with different devices and is easy to learn, with beginners reaching 52.17 words per minute on average, achieving 77% of their QWERTY typing speed with gaze tracking and 58% without.

CCS Concepts: • **Human-centered computing** → **Text input**.

Additional Key Words and Phrases: text entry; typing; gaze interaction; virtual reality, augmented reality.

ACM Reference Format:

. 2021. TapGazer: Text Entry with Finger Tapping and Gaze-directed Word Selection. In *CHI '21: ACM CHI Conference on Human Factors in Computing Systems, May 8-13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Text entry is one of the most frequent, important and demanding tasks in personal computing. Because efficient text entry methods are crucial to productivity, an enormous amount of research has been conducted on methods that improve their usability [20]. As new types of electronic devices such as smartphones have become available, new text entry methods have been proposed [22]. With the increasing popularity of consumer-grade head-mounted displays

Manuscript submitted to ACM

(HMDs) such as Virtual Reality (VR) headsets and Augmented Reality (AR) glasses, there is an expanding interest in text entry methods that can support such wearables [52]. For example, if smartphones are replaced by inexpensive and lightweight AR glasses, an AR analogue of 'texting' will be required. It is plausible that most HMDs will allow gaze tracking and finger detection in the near future in order to provide the best possible user experience; therefore a future text entry method will likely be able to take advantage of two faculties that have traditionally been used in typing: our fingers and our eyes. In order to be successful, a text entry method suitable for HMD wearers needs to address the following challenges:

Availability. HMD users are much more constrained than desktop users in what hardware they can use. Physical keyboards are often unavailable or out of reach, e.g. when users are moving freely or interacting with their real or virtual environment. In the case of AR, their environment may often be uncontrolled, e.g. on a bus, so text entry should ideally rely only on hardware that can be carried unobtrusively.

Accessibility. HMD users are constrained in their access to user interfaces. They may not be able to look at their hands or at a physical keyboard, e.g. due to occlusion from a VR headset or the need to stay aware of their environment when using AR. Therefore text entry should ideally avoid the need for the user to look at hands or manual devices. Also, users may be sitting, standing or even walking (not uncommon for mobile texting), so a text entry method should ideally be accessible from a wide range of poses.

Learnability. Because text entry is a basic task of computing systems, it should ideally be easy to learn. Because many users are already proficient in the use of a QWERTY keyboard, much previous work has aimed to exploit this familiarity to improve learnability. [8, 77].

We propose TapGazer, a new text entry method designed to address these challenges. TapGazer allows users to perform text entry simply by tapping their fingers, without needing to look at their hands or be aware of finger position. Taps may be detected with any input device capable of discerning which finger is currently being tapped, e.g. finger-worn accelerometers such as TapStrap, touch sensitive surfaces such as smart cloth, or visual finger tracking, since the location where a finger is tapped is not needed. Tracking fingers' identities and detecting whether a finger has tapped is less complicated and more accurate than tracking both the identity and location of each finger, and it is generally easier for users to focus on tapping their fingers without the need to worry about finger location. Given a suitable input device, any available surface may be used to support the hands and facilitate tapping movements, e.g. a table or one's thighs. As a result, TapGazer is designed to be beneficial not only for HMD users, but for scenarios including typing while standing, typing without a physical keyboard, and typing for mobile devices.

To enable text entry by finger tapping, TapGazer simplifies keyboard input by assigning multiple letters to each finger. Because this mapping is one-to-many, it is ambiguous (see the color-coded keyboard layout in Figure 1(b)). We resolve this ambiguity by showing word suggestions in the users' display and allowing them to select the correct word via gaze. As a result, users do not need to look at their fingers or input device. TapGazer's finger-to-letter mapping is based on a QWERTY keyboard layout. As a result, people can reuse their QWERTY skills and retain the performance benefits of ten-finger typing, which is generally faster than alternatives such as word-gesture keyboards [15]. TapGazer supports entry of unknown words, symbols and cursor navigation by allowing users to switch between different modes. Furthermore, because gaze tracking may not always be available, we describe variants of TapGazer that work without gaze tracking. We investigate the following research questions:

RQ1 How can text input be efficiently implemented using only finger taps and gaze?
RQ2 How does TapGazer perform in terms of speed, accuracy and user preference?

RQ3 How can we model user performance in TapGazer?

We address these questions by first discussing the design of TapGazer (RQ1), then reporting on a user study evaluating TapGazer (RQ2), and lastly providing a model-based analysis of how different users of TapGazer will likely perform (RQ3).

Novelty. Some previous work has looked at reduced QWERTY keyboards and word disambiguation. VType [23] applies a reduced keyboard layout, attempting to reconstruct words automatically based on finger sequence, grammar and context, but does not allow users to choose between ambiguous words. The 1Line keyboard [51] and the stick keyboard [29] flatten the QWERTY keyboard from three rows to one, allowing users to choose between ambiguous words through touchscreen gestures and arrow keys. Yet to the best of our knowledge we are the first to investigate tapping while resolving ambiguity through gaze. Also, the performance we measured for TapGazer (average 52.5 wpm with gaze tracking no completion, max. 81.3 wpm) surpasses that reported for similar works (see Table 1). Our work makes several key contributions:

(1) A design that combines tap and gaze for effective text entry, with variants for use without gaze tracking and for accommodating different user preferences.
(2) An evaluation showing that TapGazer is usable, easy-to-learn for QWERTY users, and able to reach average speeds of 52.5 wpm.
(3) A model-based performance analysis illustrating the effects of different design options and usage strategies.

2 RELATED WORK

The QWERTY keyboard is the most widely used text entry method in English speaking countries. Originally designed around the hardware limitations of early mechanical typewriters, QWERTY is not optimal for modern keyboard technology. As a result, many alternative keyboard layouts [80, 110] have been proposed to optimize typing speed or energy, e.g. ATOMIK [60] and Quasi-Qwerty [6]. For devices where a physical QWERTY keyboard is not available, many specialized text entry solutions have been proposed, e.g. for touch screens [40, 51, 83], mobile phones [22, 112], and handheld devices [13]. Moreover, using a finger [7, 73] or pen [42] for handwritten text input has been considered, although this is slow compared to typing. Speech-to-text is also a widely explored option with the potential to be faster than typing [81]; however, it has limited accuracy and is not always suitable, e.g. when the environment is noisy, other people are talking, or the content is of a sensitive or personal nature.

A key requirement of manual typing approaches – typing with hands – is detection and tracking of hands or fingers. Various methods for this have evolved over the last decades, including using gloves [50, 91], markers [33, 64], audio signals [95], cameras [105], and specific devices such as Leap Motion [104]. Moreover, various input recognition methods have been proposed, with some recognizing input as single characters ('character-level') and others recognizing entire words ('word-level'). Methods recognizing larger chunks of input (e.g. words, sentences) are typically more effective than those recognizing characters [92]. Lastly, there are good online input prediction and correction methods that can be used to improve the performance of text entry [28, 72, 111]. To develop a fast and usable text entry design using tap and gaze, we closely investigated prior work in alternative keyboard layout design across HMD and non-HMD scenarios, gaze interaction, and text entry in VR/AR. An overview of the most relevant and fastest methods, with their average speeds listed in words per minute (WPM), is shown in Table 1.

Design	Average WPMs	Examples
Tapping QWERTY on a touch surface	17.2–44.6	BlindType [55], TOAST [83], PalmBoard [103]
Gesture typing	16.0–42.7	GestureType [106], Chen et al. [15], KeyScretch [17]
Typing on tiny surface	26.0–41.0	Vertanen et al. [92], VelociTap [93], Ahn & Lee [2]
Mid-air chord gesture typing	22.0	Adhikary [1], Sridhar et al. [89]
Typing with pinch gestures	11.9–23.4	BiTipText [101], DigiTouch [97], TipText [101]
Mid-air finger tapping	17.8–23.0	ATK [104], VISAR [21]
Reduced physical QWERTY keyboard	7.3–30.0	Stick [29], 1Line [51], LetterWise [58], VType [23]
Tapping with head or controller on a soft alternative keyboard	10.2–21.1	RingText [100], PizzaText [108], HiPad [37], Boletsis & Kongsvik [8], Curved QWERTY [102]
Tapping QWERTY with head or controller	11.3–15.6	Tap/Dwell [106]
Gaze typing plus touch	14.6–15.5	EyeSwipe [46, 47], TAGSwipe [44]

Table 1. Summary of prior text entry solutions for English words that are compatible with our usage scenarios.

2.1 Alternative Keyboard Layouts

Alternative layouts generally aim to increase performance, often while supporting a limited interaction size with a reduced number of keys, which makes them relevant for TapGazer. Many layouts are designed based on optimization of typing performance, e.g. metropolis keyboard [109] and GK-D and GK-T [85]. Another common consideration is similarity to familiar layouts such as QWERTY or T9 for learnability, e.g. for mobile phones [22, 58], smart glasses [2], and smartwatches [78]. Familiar layouts are often adapted to new typing gestures, e.g. using thumb-to-finger interaction for small-screen devices or VR/AR using split QWERTY [70, 97] or T9 layouts [98]. Another trend is rearranging keyboard characters into different 2D or 3D shapes: QuikWriting [76] and its gaze-version [4] distribute letters into a circle; PizzaText [108], WrisText [27], and HiPad [37] use a pie-shaped layout; Keycube [11] attaches push buttons to a physical magic cube for typing.

When applying a reduced keyboard layout, fingers or keys are not uniquely assigned to characters, so a mechanism for disambiguation becomes necessary. LetterWise [58] uses prefix-based rather than word-based disambiguation, i.e. users press a button if the current character is wrong and then the respective character of the next-likely prefix is shown. By repeatedly pressing the button, even non-dictionary words can be typed. Stick keyboard [29] compresses the QWERTY keyboard into one line, with each key mapped to 2-3 characters. Users choose one of several ambiguous words by scrolling through possible candidates with button presses. Similarly, 1Line keyboard [51] reorganizes the QWERTY keyboard to a single line specifically for touchscreen typing, using touch gestures to support candidate selection. Similar to these works, TapGazer is based on a reduced QWERTY layout, but it uses different mechanisms for faster disambiguation.

2.2 Gaze-assisted Text Entry

Text entry with gaze does not require a physical keyboard, therefore it is a natural option to consider for HMDs, which can incorporate gaze trackers. Gaze-only methods mainly fall into the following categories [63]: direct gaze pointing with dwell ("gaze typing"), eye switches, discrete gaze gestures, and continuous gaze gestures ("gaze writing"). Dwell [5, 35, 62] (i.e. looking at keys for a certain time to trigger clicks) has been widely applied and optimized to solve the Midas Touch problem [36] (i.e. inadvertent clicks). Approaches for reducing the dwell time necessary for each key

have been explored, e.g. by dynamically adjusting it based on prefix [61], word frequency, or character placement [67]; however, it is still a major factor slowing down typing speed. Eye-switch approaches try to avoid dwell by using other operations such as blinking, brow interaction, and head movements [26] as triggers. Similarly, discrete gaze gestures have been proposed to avoid dwell, e.g. by adding a resting zone in the typing area [4], 'swiping' over a keyboard with gaze to enter a word [14, 46], or using other confirmatory eye movements such as inside-outside-inside saccades [82]. Some disambiguation algorithms have been proposed to improve the accuracy of word-level gaze gestures [53, 74]. Dasher [96] uses continuous gaze gestures to zoom towards and select candidate letters and words. Gaze-only text entry methods are much slower than typing with a keyboard; therefore some approaches try to speed it up by using other modalities for key and word selection, e.g. a physical button [45], a brain-computer interface [57], or touch gestures [2, 44]. If gaze tracking is not available, many gaze-based approaches can be modified to use head movement only [100, 106]. This can be combined with other head gestures, e.g. nodding for letter selection [54]. Overall, gaze-based text entry methods facilitate social privacy and can be used while standing or moving in VR [79]; however, they are still much slower than physical keyboards (below 25 WPM) as gaze movements that require fixations between saccades are generally time consuming (in the order of 100-400 ms [25]). Therefore, TapGazer uses gaze for disambiguation rather than typing.

2.3 Text Entry in VR/AR

Various methods have been investigated for text entry in VR/AR [18]. Because text entry using a physical keyboard is faster than other typing solutions, many approaches for text entry in VR/AR try to facilitate access to a standard physical keyboard rather than replace it. This has mainly been done by tracking and visualizing a physical keyboard in VR while sitting at a desk, either by blending in a video stream showing the real keyboard [9, 38, 52, 66] or by visualizing the keyboard in VR [9, 30, 41, 71, 94]. To support better mobility, HawKEY [77] uses a portable keyboard for users to type on while standing and walking in VR. These approaches show that using a physical keyboard and high-quality tracking leads to good performance, although the setup can be complex. However, most HMD users are unlikely to carry a full-sized physical keyboard around with them, and handling such a keyboard while moving and interacting in an uncontrolled environment can be cumbersome.

Other work has investigated VR/AR text entry with pointing gestures on virtual keyboards. Xu et al. [99] and Speicher et al. [88] compared pointing methods to selecting virtual keys with controllers, head and hand. Boletsis & Kongsvik [8] proposed virtual keyboard layouts to optimize controller-based key selection. PizzaText [108] arranges virtual keys in a circle separated into segments. Anonymous [90] compared controller-based with stylus-based virtual keyboard interaction. Yanagihara et al. [102] introduced a curved virtual QWERTY keyboard, allowing users to use a controller to swipe between different keys (21 wpm). Similarly, Chen et al. [15] proposed word gestures by pointing and swiping at a virtual keyboard. Additionally, Dube and Arif [19] researched the impact of key design on virtual keyboards for typing speed and accuracy. While these approaches improve mobility, they are much slower than physical keyboards, with typical speeds below 25 WPM.

Many VR/AR text entry methods use fingers or hands directly. A popular approach is to detect pinch gestures between fingers and thumbs, e.g. using a data glove. Pinch keyboard [10] combines pinch with hand rotation and position to select letters. KITTY [43] uses pinch gestures on different parts of the thumb. PinchType uses a reduced keyboard, and if necessary, allows the user to disambiguate words with hand gestures [24]. DigiTouch [97] uses continuous touch position and pressure. Quadmetric [48] and HiFinger [39] support one-handed text entry with pinch. BlindType [55] uses single-thumb touchpad gestures. RotoSwype [32] uses one-handed word gestures by rotating a ring worn on

one hand. Yu et al. propose one-dimensional 'handwriting' of words with a tracked finger or controller [107]. Pinch and word gesture based approaches are flexible but slow, with typical speeds far below 20 WPM. Also, mid-air finger gestures can be hard to track and can lead to fatigue when performing longer tasks [1, 21]. FaceTouch [31] allows users to type on a touch surface attached to their HMD. ARKB [49] proposes visual tracking of fingers for tapping on a virtual QWERTY keyboard. VISAR [21] facilitates mid-air one-finger tapping on an AR QWERTY keyboard. VType [23] uses finger tapping on a reduced QWERTY keyboard layout and reconstructs words based on finger sequence, grammar, and context for text input in VR. The accuracy reported for a predefined vocabulary is high; however, no method for disambiguation between candidate words was considered and no typing speed was reported. Tapping on a reduced QWERTY keyboard is promising for text entry in VR/AR as it is flexible and robust compared to alternatives. Therefore, we explore how it can be optimized by using gaze input and additional taps for disambiguation.

3 TAPGAZER DESIGN

TapGazer allows users to tap words as if they are typing them on a physical QWERTY keyboard, and then disambiguate their tap input selecting the desired word through gaze. It was designed primarily for HMD users, but could also be useful for other scenarios where more conventional input devices are unavailable or difficult to access. Given suitable sensors, users can type by tapping their fingers on any surface or even in mid-air. As TapGazer only considers the identity of the finger that is currently tapping and not its position, it only needs to know which of the user's 10 fingers has just been tapped, if any, at any given time. Each of the 26 letters of the alphabet is mapped to one of the eight non-thumb fingers, while the two thumbs are reserved for controlling use including word selection, deletion, etc. Figure 1 illustrates the state machine of TapGazer with gaze selection and word completion. Starting from an idle state, TapGazer waits for tap or gaze input events. Except for the thumbs, a finger tap adds a letter to the *input string*, starting from an empty string. The input string is constructed from an *input alphabet* with one character for each of the eight fingers: we are using the characters `asdfjkl;`, which correspond to the rest positions of each finger on a QWERTY keyboard, for later reference. When typing a word with TapGazer, the user taps the fingers as they would do when typing on a QWERTY keyboard. However, as each finger tap can be interpreted as one of several characters, the word represented by the input is ambiguous: for example, `fjd` is the input string for the words 'the' and 'bye'. We refer to a set of words that all have the same tapping input string as a *homograph set*. A tap with the left thumb deletes the current input string so users can start the word again. A tap of the right thumb selects the word to enter from a list of suggestions while the word is pointed at by the user's gaze.

As a user enters an input string, the central area of TapGazer's user interface shows a list of word *candidates*: similar to predictive text on a mobile phone, the user is given a list of the most likely words to choose from. TapGazer shows all words in the *homograph set* for the given input string, which we call *complete candidates* as they are based on the whole input string (e.g. 'the' for `fjd`). Additionally, TapGazer uses a language model to show the most likely *incomplete candidates*, i.e. words with a prefix matching the current input string (e.g. 'these' for `fjd`). After each tap, TapGazer updates the candidates shown. In order to select a candidate, the user looks at it, and in response the fixated candidate is highlighted with an underline. If the right thumb is tapped, the currently highlighted candidate is selected and added to the entered text. At this point, the TapGazer state machine starts again with an empty input string. If the user taps the right thumb but does not fixate any candidate, then the most likely candidate is selected based on a language model. Figure 2 illustrates how to type 'chi' with TapGazer. Word completion in TapGazer can be disabled; in this variant only complete candidates are shown if they exist. If no complete candidate exists, we show the shortest incomplete candidate to inform users about the progress of typing. Furthermore, we have designed a purely manual variant of TapGazer

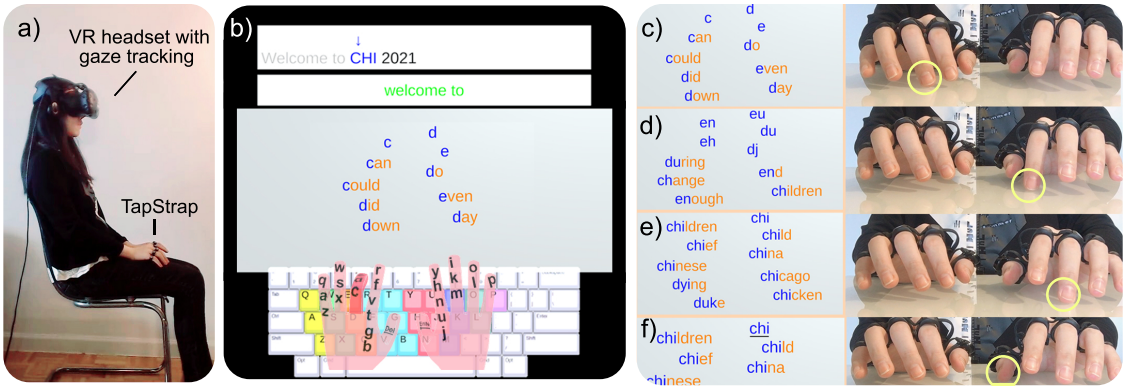

Fig. 2. Text entry example: a) A user is 'typing' on her thighs using a TapStrap device instead of a keyboard. b) The user just started to 'type' the word "CHI". The interface provides optional visualizations of the finger-key mapping as a virtual keyboard and/or hands. c) The user first tapped the left middle finger, which is mapped to 'c', then d) the right index finger, and e) the right middle finger. f) Finally, the user looks at the word "chi" (underlined because of being gazed) and taps the right thumb to select the highlighted word.

without gaze tracking, allowing users to disambiguate candidates with extra taps. Figure 4 illustrates different input devices and variants of Tapgazer.

In answering RQ1, several design decisions were made: First, we use finger tapping so that users can 'type' on any surface and require no context knowledge between the surface location and finger/hand location. Second, we help users find the word to type in the list of candidates, by facilitating visual search in the layout of the graphical interface. Third, we provided word completion and compare whether word completion benefits TapGazer in terms of performance.

3.1 Virtual Keyboard Layout

Customization. TapGazer reuses the standard QWERTY layout to avoid an extra learning curve from new keyboard layouts. However, in our pilot studies we found people have varying finger preferences for typing on the QWERTY keyboard, e.g. key 'b' may be pressed with either the left index finger or the right index finger. As a result, TapGazer creates a profile for each user to record their finger-to-key mapping. To guide novice users, we optionally visualize the customized finger-to-key mapping in a virtual keyboard and/or a hand model (Figure 2b), with each key colored according to its associated finger and letters rendered on their corresponding fingers. We use prefix trees to quickly look up complete and incomplete candidate words and their word frequencies for each input string, which are generated based on the users' mapping.

Feasibility. Text entry is only feasible if all the words in the homograph set of any input string can be somehow selected. The *minimum candidate number* (MCN) is the minimum number of candidate words the interface must be able to disambiguate at a time. It is equal to the maximum number of homographs an input string can have, i.e. it describes the worst possible ambiguity that may need to be resolved. It is important for the design to determine the MCN in advance because display space needs to be adequately allocated, or users must be given the option to page through sets of candidates. The MCN is also relevant for performance as it describes the worst case scenario of visual search for the right candidate. We determined popular QWERTY-based finger-to-key mappings in pilot experiments and then ran a simulation to determine their overall MCN based on different word sources: the 1000 most common words retrieved from Wikipedia with $MCN_{1K} = 4$; the standard MacKenzie phrase corpus [59], which contains 500

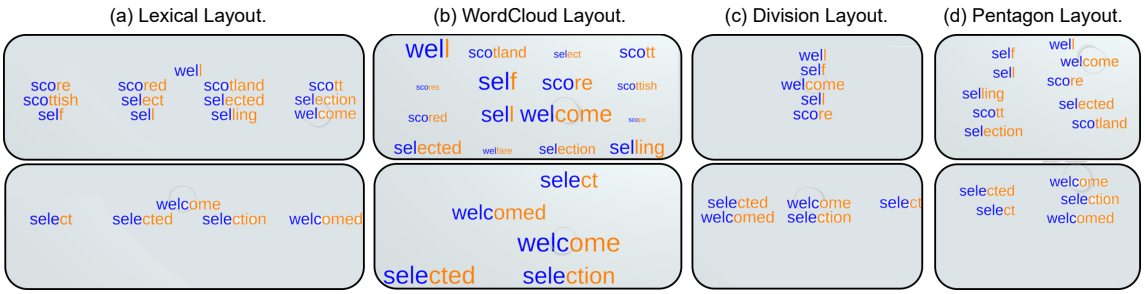

Fig. 3. Evolution of TapGazer layout designs: a) *Lexical Layout* places the most common candidate word in the first row and arranges the other candidates in alphabetical order. All the candidates have the same font size. b) *WordCloud Layout* emphasizes frequent candidates with a larger font size. Candidates that were already shown on the previous tap keep their position. c) *Division Layout* divides all candidates into three columns according to their last letter. d) *Pentagon Layout* orders the candidates based on frequency and arranges the candidates in a compact single or double pentagon shape.

phrases for evaluation use, with $MCN_{MacKenzie} = 6$; and the 90% most frequent words (7,440) generated from the wordfreq library [87], which includes many very-low-frequency specialized words and acronyms that are not typically part of dictionaries, with $MCN_{7K} = 7$. We design our interface to be able to show at least 10 candidates in order to cover all English dictionary words and also many low-frequency non-dictionary words across typical QWERTY finger-to-key mappings. For unsupported words such as neologisms and special acronyms, we provide a *spelling mode* for character-level text entry (see subsection 3.4).

Alternative Layouts. We also calculated the MCNs of standard keyboard layouts other than QWERTY to gauge their suitability for use in TapGazer. Optimal word gesture keyboards such as GK-D ($MCN_{MacKenzie} = 11$, $MCN_{7K} = 12$) and GK-T ($MCN_{MacKenzie} = 7$, $MCN_{7K} = 17$) [85] have higher MCNs, probably because they are not optimized for key-based typing. If the left thumb is used for tapping instead of deletion (e.g. by triggering deletion with a chord), having 9 fingers to tap reduces ambiguity in the finger-to-key assignment, potentially decreasing the MCN. We calculated the MCN for some known 9 finger layouts: standard T9 [98] ($MCN_{MacKenzie} = MCN_{7K} = 5$); HiFinger [39], which distributes letters in lexical order over nine keys ($MCN_{MacKenzie} = 5$, $MCN_{7K} = 8$); and the quadmetric optimized layout [48] ($MCN_{MacKenzie} = MCN_{7K} = 4$). Finally, we performed an extensive combinatorial search of non-QWERTY layouts and found that there is a very large number of mappings for eight fingers with $MCN_{MacKenzie} = MCN_{7K} = 4$. These results suggest that layout optimization can help to reduce the number of candidates that have to be shown at one time, which could speed up text input.

3.2 Word Candidate Layout

The most important part of TapGazer's visual interface is the central gray area where word candidates are shown for selection by the user (Figure 2b). These candidates are colored to indicate the tapping progress of each word: the prefix of each word that has already been tapped is colored in blue, while yet-to-be-tapped postfixes are colored in orange. Complete candidates are completely blue and are always shown in the interface as they must be available as word choices. Any further available space can be filled with incomplete candidates, indicating options for word completion. The number of candidates shown is a trade-off between saving taps through word completion, and visual search time spent looking for the right candidate. Visual search time is affected by the way we arrange the candidates, therefore

we designed, tested and re-designed the layout to reach a suitable design. Figure 3 illustrates the design evolution of TapGazer's candidate layout.

Initial Design. We first designed (a) Lexical Layout and (b) WordCloud Layout based on the following design principles. *Systematic locations*: Users should intuitively know where to look for a word. *Salience*: More likely words should be more salient (e.g. larger or more central). *Continuity*: Avoiding changes in the position of a suggested word between taps may help users to spot it. Lexical Layout places the most frequent word into the first line and the other candidates in alphabetical order below. This prioritizes systematic locations over continuity, as candidates' positions may change between taps, e.g. "welcome". WordCloud Layout arranges candidates in word-cloud style, with more frequent words arranged at the center and in a larger font. Candidates keep their positions between taps, prioritising continuity over systematic locations. To understand the effects of the layouts and their design principles on novices, we conducted a formative study with 12 participants (5 female, 7 male; aged 18 to 30, $M = 24.67$, $SD = 3.94$), comparing the two layouts in a within-participant design. Each participant used each layout twice for 5 minutes each time, followed by quantitative and qualitative questionnaires collecting their feedback on each layout and design principle. The quantitative results showed that there were no significant differences in typing speed in WPM ($F(1, 11) = .81, p = .39$) and scale of usability scores (SUS) [3] ($F(1, 11) = 2.1, p = .15$), but Lexical layout achieved better accuracy TER% [86] ($F(1, 11) = 7.1, p = .004^{**}$) and lower NASA-TLX task load scores [34] ($F(1, 11) = 14.32, p < .001^{***}$). Participants were split half-half in their layout preference. They immediately understood Lexical Layout's systematic locations, but did not find them very helpful. Having the most frequent word at the top or centre was found useful, but variations in font size were found to be distracting when typing low-frequency words. Some participants noticed WordCloud's continuity, but did not find it very helpful as tapping is too fast to visually follow candidates. This was an important lesson: It is not useful to design the visual layout around the tapping process, as fingers are much faster than the eye. It is more useful to consider the layout as a pure visual search task, where visual search time is correlated with the number of candidates and the distance of eye movement [69].

Final Designs. Division Layout (Figure 3c) distributes candidates into three columns according to their last letter, ordering each column by word frequency. The column boundaries were chosen to balance the expected number of candidates in each column, with words ending in A-E on the left, F-R in the middle, and S-Z on the right. This layout is designed for power users who have learned where to expect a word, potentially reducing search time by 2/3. Pentagon Layout (d) is designed to be suitable also for novices. It arranges candidates in compact groups of five, close together to minimize eye movement but with enough separation for accurate gaze selection (at least 0.5° visual angle). The pentagon shapes mitigate overlap between long adjacent words and try to take advantage of people's ability to quickly scan groups of five items at a time [65]. Complete candidates are always shown before incomplete candidates, with frequent words closer to the top. We chose Pentagon Layout for our main study as it is easier to use for non-experts.

3.3 Disambiguation

After presenting possible word candidates, users need to select a candidate to disambiguate the input. In text entry on mobile devices, word candidates are commonly selected by touch and users typically first look at the candidate they want to select, which is a subliminal step. TapGazer takes advantage of this by employing gaze tracking for word selection with the aim of minimizing taps and reducing cognitive load. Once the user has found the right word, she can select it with a tap of the right thumb. We chose to use a tap rather than a gaze-dwell for selection as the latter is much more time consuming and can lead to Midas Touch (inadvertent activations) [75].

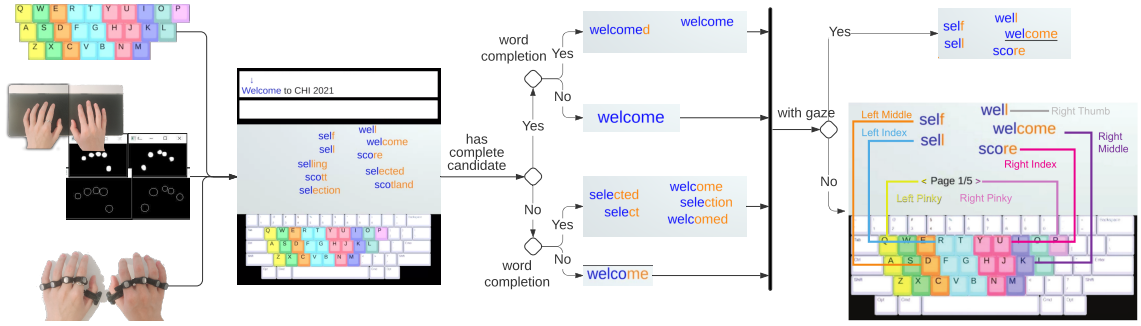

Fig. 4. TapGazer's workflow: After receiving a tap from a suitable input device, TapGazer updates the candidates according to the word completion mode, and allows users to select a candidate either with gaze or with additional taps.

In the absence of gaze tracking, we provide a variant of TapGazer with purely *manual selection* (Figure 4 bottom-right). In this variant, selecting a candidate is a two-step operation: 1) tapping with the right thumb, and 2) tapping with one of five fingers (right thumb, right middle, right index, left middle, left index) to select one among a maximum of five candidates shown. In order to support selection from more than five candidates, users can page through sets of five candidates with their left and right little fingers. The layout design helps to avoid paging operations by showing complete candidates first, and ordering complete and incomplete candidates by their descending word frequencies.

3.4 Miscellaneous Text Entry Functionality

We have designed miscellaneous text editing functions for TapGazer in order to make it a complete text entry method. *Deletion* of the current input string is performed by tapping the left thumb, allowing users to start a word again. If left thumb is pressed right after selecting a candidate, the candidates for the last input string will be shown again, allowing users to change the selection or tap left thumb again to delete the word. *Spelling mode* is triggered with a chord operation. Users can switch between word-level and character-level text entry by tapping their left and right index fingers simultaneously. Afterwards, users can rotate through the characters mapped to each finger by repeatedly tapping a respective finger, and enter the character by tapping the right thumb. Tapping the right thumb again concludes the character-level input. *Cursor navigation* with gaze is performed by selecting words in the entered text directly with gaze and right thumb [84], or by entering a cursor navigation mode though a button in the periphery of the interface [56] with gaze and right thumb. Users can then move the cursor by tapping the left/right index finger and exit cursor mode with a right thumb tap. If gaze is unavailable, users can enter cursor mode by tapping the right index and ring fingers simultaneously.

3.5 Design Limitations

We observed that some QWERTY users occasionally use different fingers for the same key. TapGazer currently does not support this behavior as finger-to-key mappings only allow keys to be mapped to a single finger. Many-to-many mappings are in principle possible, but would likely increase the MCN and the visual search time. Word prediction is currently based only on word frequency; it could be substituted by a more advanced method taking also the context of a word into account. We currently do not provide auto-correction, as this could confound our study of accuracy.

4 EVALUATION

We conducted a user study in order to examine how the different TapGazer variants perform in terms of speed, accuracy, and user preference (RQ2) as compared to a physical QWERTY keyboard. We used a within-subject design with five conditions: standard QWERTY keyboard (K), TapGazer with gaze candidate selection and word completion (GC), TapGazer with gaze candidate selection and no word completion (GN), TapGazer with manual candidate selection and word completion (MC), and TapGazer with manual candidate selection and no word completion (MN). The order of the five conditions was counterbalanced to mitigate effects of learning and fatigue.

The study had to be conducted remotely using videotelephony. Participants were using their personal computers running Windows or MacOS, so participants were tapping on their keyboards when using TapGazer. Participants were free to decide where on the keyboard they wanted to tap each finger, i.e. which fingers would tap which keys. We helped them to assign keys along the natural ranges of motion of each of their fingers, so that they could comfortably tap their fingers without having to consider the keys. Equally, TapGazer did not consider individual keys but only used the keyboard to identify which finger was tapped. As most participants did not have access to gaze trackers, we impressed on them to locate the right candidates with their eyes as a simulation of using a gaze tracker in GN and GC. If the right word was shown in the candidate area, we assumed that participants selected the correct word when tapping with the thumb. If the right candidate was not shown, this would result in an incorrect word; so in line with real gaze racking, participants had to first spot the right candidate to ensure a correct input was made.

Measures. We compared the conditions using objective measures [86] such as typing speed (WPM), total error rate (TER) and average times of frequent operations, as well as subjective measures such as SUS usability scores [3] and NASA-TLX task load scores [34]. Each tap/keystroke operation was recorded for analysis.

Procedure. After a brief introduction of TapGazer, participants gave consent to join a video call and share their screen during the experiment. First, we measured their QWERTY typing behavior and generated their customized virtual keyboard for TapGazer. Then they performed each of the five conditions (K, GC, GN, MC, MN) in counterbalanced order, with each condition consisting of a training session and five test sessions. In the training session, participants received brief instruction on the use of the respective text entry method and were able to practice it for a couple of minutes. In the five test sessions participants were asked to enter phrases randomly sampled from the MacKenzie & Soukoreff corpus [59], as fast and accurately as possible while looking at the screen. Each test session was one minute long and participants were allowed to take short breaks between the sessions. After each condition, participants completed SUS and NASA-TLX questionnaires. Each condition took around 10-15 minutes. After finishing all five conditions, participants completed a demographics questionnaire and ranked the conditions and their selection and completion modes according to their preference. Lastly, participants were interviewed about TapGazer, the reasons for the rankings, and their general suggestions. On average, the experiment took about 60 to 90 minutes.

Participants. We recruited 20 participants (3 female and 17 male; aged 24 to 35, $M = 28.25$, $SD = 3.45$) via word of mouth and flyers. 15 had used eye-tracking devices before. The average typing speed was 67.13 WPM with a TER of 8.47% ($SD = 0.025$) on a standard QWERTY keyboard.

4.1 Results

We validated that the data satisfies the assumptions of a repeated measures analysis of variance (ANOVA). We used one-way repeated measures ANOVAs to compare effects across all five conditions and their sessions, and two-way repeated measures ANOVAs to compare the effects of the different TapGazer variants with regards to the factors Gaze

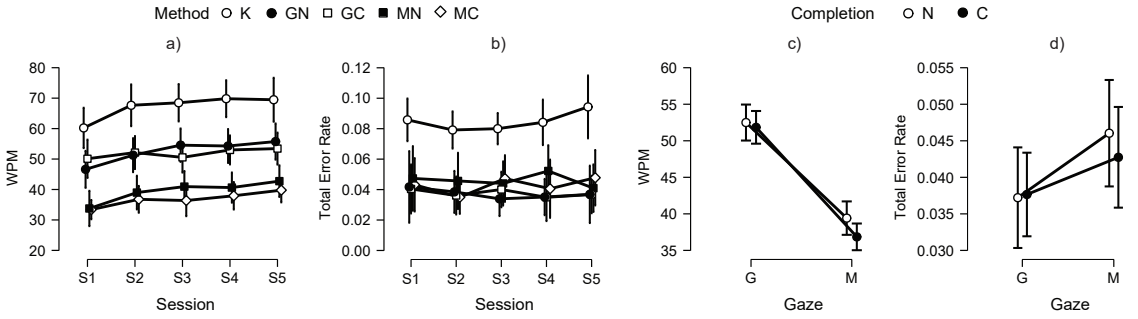

Fig. 5. Evaluation results for WPM and TER comparing QWERTY keyboard (K) and TapGazer in GN, GC, MN, and MC modes.

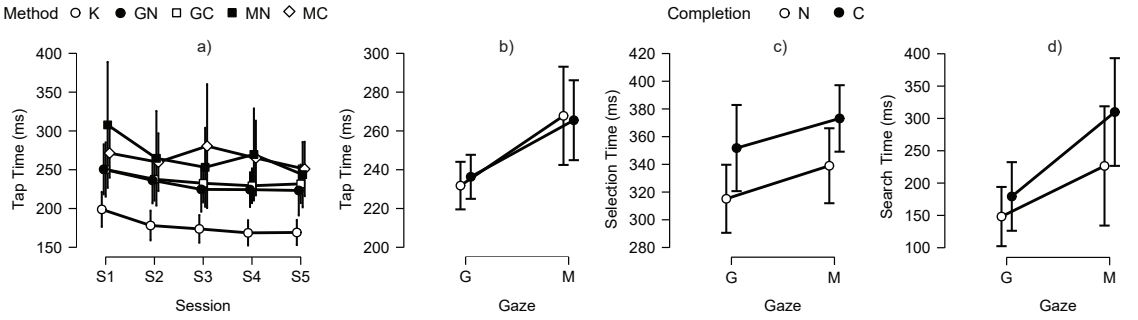

Fig. 6. Evaluation results for operation times comparing QWERTY keyboard (K) with TapGazer in GN, GC, MN, and MC modes.

(gaze vs. manual) and Completion (word completion vs. no word completion). Paired t-tests with Holm correction were used for all pairwise comparisons between conditions. All tests for significance were made at the $\alpha = 0.05$ level. The error bars in the graphs show the 95% confidence intervals of the means.

Text Entry Speed. Figure 5a shows the average text entry rate for the five conditions. Users typed at $M = 52.49$ ($SD = 13.44$) for GC, $M = 51.84$ ($SD = 8.92$) for GN, $M = 39.42$ ($SD = 10.75$) for MC, and $M = 36.85$ ($SD = 8.47$) for MN. The main effects of Condition ($F(4, 76) = 69.56, p = .001^{***}$) and Session ($F(4, 76) = 19.81, p < .001^{***}$) were both significant. K was significantly faster than all TapGazer variants ($t(18) \geq 7.13, p < .001^{***}$). For TapGazer (Figure 5c), the main effect of Gaze ($F(1, 19) = 74.98, p < .001^{***}$) was significant with a 'large' effect size ($\omega^2 = 0.37$), indicating that gaze selection was faster. The main effect of Completion ($F(1, 19) = 1.517, p = .23$) and the interaction effect ($F(1, 19) = 1.92, p = .18$) were not significant.

Tap Time. Figure 6a shows the average times per tap when entering letters of a word. The main effects of Condition ($F(1, 19) = 13.71, p < .001^{***}$) and Session ($F(1, 19) = 7.11, p < .001^{***}$) were both significant. Tapping with K was faster than for all TapGazer variants ($t(18) \geq 3.89, p \leq .002^{**}$). For TapGazer (Figure 5b), the main effect of Gaze ($F(1, 19) = 7.84, p = .012^{*}$) was significant with a 'small to medium' effect size ($\omega^2 = 0.04$), indicating that taps with gaze were faster. The main effect of Completion ($F(1, 19) = 0.01, p = .92$) and the interaction effect ($F(1, 19) = 0.28, p = .61$) were not significant.

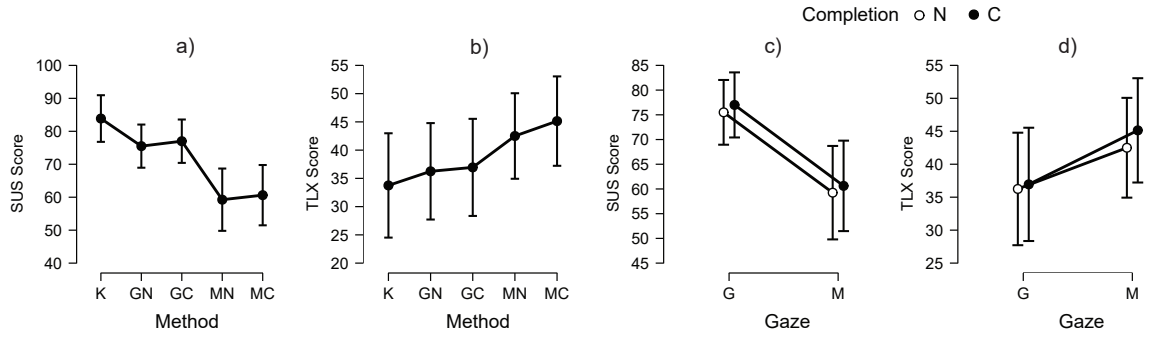

Fig. 7. Evaluation results for SUS and TLX comparing QWERTY keyoard (K) with TapGazer in GN, GC, MN, and MC modes.

Selection Time. Figure 6c shows the average the average times taken to select a candidate in Tapgazer after tapping a word's input string. The main effects of Gaze ($F(1, 19) = 1.19, p = .29$) and Completion ($F(1, 19) = 4.37, p = .05$) and the interaction effect ($F(1, 19) = 0.01, p = .93$) were not significant.

Search Time. Figure 6d shows the average times taken between tapping a word's input string and confirming the choice of a candidate with a tap, approximating visual search and think time by not counting the taps involved in selection. Gaze selection requires an additional right thumb tap, and manual selection requires an additional right thumb tap and a thumb/index/middle finger tap, which were adjusted for by subtracting the respective participant tap time averages. The main effects for Gaze ($F(1, 19) = 7.33, p = .014^{*}, \omega^2 = 0.11$) and Completion ($F(1, 19) = 7.99, p = .01^{**}, \omega^2 = 0.05$) were significant with 'medium' effect sizes, indicating that candidate search was faster with gaze selection and without word completion, respectively.

Error Rate. Figure 5c shows the average total error rate for the five conditions. Users typed at $M = 8.47\%$ ($SD = 0.025$) for K and $M \leq 4.60\%$ for TapGazer. The main effect of Condition ($F(4, 76) = 29.85, p < .001^{***}$) was significant, and the main effect of Session ($F(4, 76) = 1.24, p = .30$) was not significant. All TapGazer variants had significantly lower error rates then K ($t(18) \geq 8.14, p < .001^{***}$). For TapGazer (Figure 5d), the main effect of Gaze ($F(1, 19) = 3.16, p = .09$) and Completion ($F(1, 19) = 0.20, p = .66$) and the interaction effect ($F(1, 19) = 0.23, p = .64$) were not significant.

Usability. Figure 7a shows the average SUS usability scores for the five conditions, with $M = 83.88$ ($SD = 15.12$) for K, about $M = 78$ ($SD = 14$) for GN and GC, and about $M = 61$ ($SD = 19.55$) for MN and MC. TapGazer with manual selection had significantly lower SUS scores then the other conditions ($t(18) \geq 3.75, p \leq .002^{**}$), and the differences between TapGazer with gaze selection and K are not significant ($t(18) \leq 2.1, p \geq .15$). For TapGazer (Figure 5c), the main effect of Gaze ($F(1, 19) = 24.00, p = .001^{***}$) was significant. The main effect of Completion ($F(1, 19) = 0.47, p < .50$) and the interaction effect ($F(1, 19) = 0.001, p = .97$) were not significant.

Workload. Figure 7b shows the average NASA-TLX task load scores for the five conditions, with $M = 33.75$ ($SD = 19.73$) for K, about $M = 36$ ($SD = 18$) for GN and GC, $M = 42.50$ ($SD = 16.18$) for MN, and $M = 45.14$ ($SD = 16.91$) for MC. MC had significantly higher task load scores than K ($t(18) \geq 3.18, p \leq .02^{*}$); the other differences are not significant ($t(18) \leq 2.48, p \geq .14$). For TapGazer (Figure 5d), the main effect of Gaze ($F(1, 19) = 6.23, p = .02^{*}$) was significant with a 'medium' effect size ($\omega^2 = 0.04$), indicating that gaze selection has a lower task load. The main effect of Completion ($F(1, 19) = 1.00, p < .33$) and the interaction effect ($F(1, 19) = 0.28, p = .61$) were not significant. We analysed the six

dimensions of task and in NASA-TLX separately and found that the only significant effect was the main effect of Gaze on Frustration ($F(1, 19) = 11.98, p = .003^{**}$), indicating that gaze selection was less frustrating.

Preference and Qualitative Feedback. When asked about which variant of Tapgazer they preferred, 15 of the 20 participants preferred GC the most and 5 preferred GN the most. In the post-interviews, participants were overall positive about Tapgazer ("*I can type very fast after practice*", "*save energy by just tapping without reaching the specific letter*"). Several participants stated it was easy to find the right candidate words ("*the candidate words are different from each other and easy to locate the word to type*", "*look at where the word will show up and select it when it shows*"). Several participants noted that gaze selection was more usable than manual selection ("*it is really convenient when using gaze mode*", "*selecting words with gaze is more comfortable. It is easy to make mistakes while using manual selection mode*", "I need to think about which finger to tap when using manual selection"). Most participants appreciated the ability to complete words ("I'd love to have more words to select from", "*can save quite a few keystrokes when using TapGazer with completion*", "*I want to scan all candidates words in case I found the correct one*"), but some noted that it was easier not to consider completion of words ("*focus on typing the word and no need to worry about the candidates shown. And usually I don't need to type the entire word when it is long*" – referring to the fact that in the TapGazer variants without word completion, the most likely incomplete candidate is shown in absence of complete candidates, allowing users to quickly complete long words). All participants reported that they would be willing to use TapGazer for off-desktop scenarios like VR.

4.2 Discussion

Our results demonstrate that overall, TapGazer is an easy-to-use system that can reach higher average WPM speeds than other text entry methods addressing similar use cases if gaze selection is used (see Table 1, on average 52.17 WPM as opposed to 44.6 for the best competitor). Furthermore, TapGazer achieves significantly lower error rates than the QWERTY keyboard, as word-level text entry avoids some sources of error of character-level text entry: while QWERTY typing, participants frequently used the right finger on the wrong key – a mistake that does not affect TapGazer. The higher WPM averages listed in Table 1 are mainly for experienced 'expert' users; however, our participants were all novice users of TapGazer. There were significant trends of improvement over the sessions, indicating that an even better performance could be achieved with more practice. TapGazer with manual selection performed markedly worse; the extra tap required for selection slightly breaks the normal rhythm of QWERTY typing and forces participants to think about finding their word as well as using the right finger for selection. However, it is still very competitive compared to the related works in Table 1, and with practice using the right finger to select the appropriate candidate will likely become automatic for a user. Tapgazer with and without word completion perform similarly and both have their place; some users prefer to just type and not think about completion of words, while others prefer to look for incomplete candidates before completing an input string. We analyse these two strategies further in subsection 5.2.

Limitations. The global COVID-19 crisis forced us to perform TapGazer's evaluation remotely. This necessitated three compromises: 1) Tapping was performed on a physical keyboard as opposed to a passive touch surface, providing active feedback through the mechanical keys; 2) although TapGazer was designed for HMD users, our participants did not have HMDs; and 3) most participants did not have access to a gaze tracker, so that we had to simulate selection with gaze. In order to address the first point, we performed a pilot study comparing TapGazer performance on a physical keyboard vs. on a passive touch surface (see subsection 5.1). The results indicate that the differences between the two are minor, so our results do likely generalize to tapping on other surfaces. In order to address the lack of HMDs, we went through the webcam footage recorded from all experimental sessions, and confirmed that users did indeed keep

looking at the screen while typing, as instructed, and not at their hands or elsewhere. This indicates that they were visually immersed in the TapGazer interface, in keeping with the experience one would expect in an HMD. In order to address the validity of simulating gaze for candidate selection, we analysed candidate selection in more detail: First, the difference in selection time for gaze vs. manual was not significant, so it is unlikely that participants would have gained a lot by 'cheating' and not looking at the right candidate. Half of the TapGazer data we collected is based on manual selection, and the manual selection time results for word completion vs. non-completion are consistent with those of gaze selection (roughly the same difference); we would have expected word completion to be more affected by 'cheating' as it would have provided more opportunities. Finally, 'cheating' would have resulted in a higher error rate for gaze, especially in GC, as participants not looking for the right candidate would likely have selected before the right candidate was shown. The tap logs confirm that participants used word completion frequently in GC, and that they would have made a lot more errors if they had simply tapped the right thumb without making sure the right candidate was present. The error rate is overall very low, and almost the same for GC and GN, which supports the assumption that our simulation of gaze was accurate. Another limitation is the fact that our study uses novices and is mainly cross-sectional, thereby likely underestimating the average performance of longer-term users. Finally, the study evaluates TapGazer only with participants seated at a desk, while our design aims for TapGazer to be usable in a variety of poses, e.g. standing. However, our setup is similar to the way most related works evaluated their performance (Table 1), so this makes our results more comparable to those of others.

5 MODEL-BASED ANALYSIS

In the following we will discuss models describing the user performance of TapGazer and its variants (RQ3) and then apply them to the analysis of design options and usage strategies. Because TapGazer is based on QWERTY typing, it is plausible to estimate performance based on a user's QWERTY typing speed. Based on the data from Section 4, wpm_K is significantly positively correlated with WPM_{GN} ($r(18) = 0.66, p < .001^{***}$), WPM_{GC} ($r(18) = 0.71, p < .001^{***}$), WPM_{MN} ($r(18) = 0.53, p = .008^{**}$), and WPM_{MC} ($r(18) = 0.58, p < .004^{**}$), with 'large' effect sizes [16]. Linear slope regression analysis yielded significant regression equations: $WPM_{GN} = 0.77 \times wpm_K$, $WPM_{GC} = 0.76 \times wpm_K$, $WPM_{MN} = 0.58 \times wpm_K$, and $WPM_{MC} = 0.54 \times wpm_K$. This confirms that there is a strong linear relationship between QWERTY typing and TapGazer performance, with users that are fairly new to TapGazer achieving on average 77% of their QWERTY typing speed when using the GC variant. Similarly, the average time taken for typing a key $type_K$ is significantly positively correlated with the average times for tapping a finger tap_{GN} ($r(18) = 0.80, p < .001^{***}$), tap_{GC} ($r(18) = 0.84, p < .001^{***}$), tap_{MN} ($r(18) = 0.76, p < .001^{***}$), and tap_{MC} ($r(18) = 0.80, p < .001^{***}$). Linear slope regression analysis yielded significant regression equations: $tap_{GN} = 1.31 \times type_K$, $tap_{GC} = 1.32 \times type_K$, $tap_{MN} = 1.52 \times type_K$, and $tap_{MC} = 1.48 \times type_K$.

5.1 Input Device

Our evaluation asked users to tap on a physical QWERTY keyboard, giving them flexibility as to which keys they tap by assigning keys to fingers according to their fingers' typical ranges of movement. However, TapGazer was designed to work without a physical keyboard, allowing users to tap on any surface. Therefore we need to analyse in how far TapGazer performance changes if, instead of a physical keyboard, we use an input device that can pick up finger taps on a surface directly. Such a device could use sensors in the surface (e.g. adequately-sized touchpads or touchscreens), sensors on the fingers (e.g. TapStrap), sensors along the neural or muscular pathways controlling the fingers (e.g. wrist EMG), or optical tracking (e.g. Leap Motion).

In order to model the effect of using a flat-surface tap input device on TapGazer performance, we conducted a small study (n=6) comparing TapGazer typing skills (using gaze and word completion) on a physical QWERTY keyboard vs. touchpad surfaces (two Sensel Morphs). In both conditions, participants wore a Tobii HTC Vive Devkit VR headset to simulate a realistic use case for TapGazer (Fig. 1a). Participants were seated at a desk and first performed a 1-minute standard QWERTY typing test. Then participants put on the headset, calibrated the gaze tracker, and ran the official Tobii gaze demo to get used to the headset. They were then shown how TapGazer works and given time to practice it until they were able to correctly type at least 5 phrases taken from the MacKenzie corpus [59] . During this training phase, we made sure that TapGazer was configured according to the finger-to-key mapping they preferred. Next, the two 5-minute typing sessions were performed, one for TapGazer on a physical keyboard and one for TapGazer with the Sensel Morphs. Participants were able to move their fingers freely on the Sensel Morph touch surface and we were able to identify each finger by its relative position. A short break was allowed between the conditions, the gaze tracker was recalibrated for the second condition, and the order of conditions was counterbalanced. Our participants (2 female, 4 male) varied in their QWERTY touch typing abilities (wpm range 28.6 - 72.5, average 50.8 wpm), two were wearing glasses, and half of them had never used eye-tracking devices before. We used correlation and regression analysis to build and validate a linear model of TapGazer speed for a touch surface WPM_{TS} based on TapGazer speed for a physical keyboard WPM_{TK}. wpm_TS is significantly positively correlated with WPM_{TK} ($r(4) = 0.86, p = .01^*$) with 'large' effect size. Linear slope regression analysis yielded a significant regression equation: $WPM_{TS} = 0.98 \times wpm_{TK}$. This indicates that the speed of tapping on a keyboard (as opposed to typing on a keyboard) is comparable to the speed of tapping on a passive surface, so it is likely that our evaluation results can be generalised to other suitable input devices.

5.2 Slow Typists

Some people are slow typists, e.g. when they are just learning to type. Word completion can be particularly useful for them. This is similar to text entry on a mobile phone, where tapping individual keys can be slow and many people use word completion extensively to speed up text input. In the following we show how to estimate the QWERTY typing speed that marks the point in typing and tapping skill where not using word completion becomes faster than using word completion with a visual search for the correct word after every tap.

Similar to the well-known Keystroke-Level Model (KLM) [12], we model the time T_{GN} required for entering a word w in TapGazer without word completion based on: a) the average tapping time for fingers tap; b) the average tapping time for the right thumb $space$ (which types space in the standard QWERTY mapping), and c) the average visual search time $search_{GN}$ for finding a desired word among the completed words shown: $T_{GN}(w) = |w| \times tap + search_{GN} + space$. That is, we sum up the average tapping time for each of the $|w|$ letters, the average search time for spotting the right completed word, and the average time of the confirmatory tap with the thumb. Note that by definition, this model predicts the average word completion time for our evaluation of GN exactly when substituting our measured average values for the model parameters. Similarly, we model the time T_{GC} required for entering a word w in TapGazer with word completion, assuming that the user looks at the suggested words after every tap. This time we consider the number of taps $c(w) \leq |w|$ required until w appears in the list of suggestions, and the average visual search time $search_{GC}$ for finding a desired word among suggested, possibly incomplete words: $T_{GC}(w) = c(w) \times (tap + search_{GC}) + space$. The model illustrates the trade-off between a reduced number of taps and increased time spent per tap.

In Section 5 we have shown that there is a strong linear relationship between tapping and QWERTY typing speed. Therefore, in order to estimate T_{GN} based on the time T_K required to type word w, we substitute tap and $space$ by corresponding linear estimates $1.31 \times type_K$ and $0.91 \times type_K$, respectively. Because search times do not vary with

QWERTY speed, we approximate them using averages $search_{GN} = 148$ ms and $search_{GC} = 420$ ms. The latter is the average for GC when the maximum number of candidates is shown (10) so that it is not immediately apparent which candidate to choose, as this is the most likely case when looking for word completions after every tap. Furthermore, we substitute the average word length in English text $|w| = 4.79$ [68], and the expected number of taps $c = 2.41$ required before a desired word appears in the suggestions. The latter was determined using a simulation of the word-frequency based suggestion algorithm used in GC on a dictionary of the 7,582 most frequent English words. This results in estimates of the average times per word dependant on $type_K$: $T_{GN} = 7.18 \times type_K + 148$ ms and $T_{GC} = 4.07 \times type_K + 1012$ ms. T_{GN} and T_{GC} are equal at $type_K = 278$ ms, which is equivalent to about $wpm_K = 37$. Therefore, typists much slower than that would likely be faster using TapGazer with word completion. A better word prediction algorithm will reduce the expected value for $c(w)$, increasing the estimated speed at which word completion becomes a hindrance. A similar analysis can be made for the non-gaze alternatives of TapGazer MN and MC.

5.3 Power Users

If the prediction algorithm used to generate suggestions for word completion is reasonably stable, i.e. if users can anticipate which word will be suggested as the most likely option, then power users will learn for frequent words how many taps they need before they can simply accept the most likely suggested word. In both GC and MC, the most likely suggestion can be quickly accepted without even looking at the word suggestions, by tapping the right thumb. Let us assume a power user has learned all the prefixes that must be tapped to make each of the 100 most frequent words of the English language the most likely suggestion, e.g. "tapping 't' makes 'the' the most likely word." According to our word frequency data, the 100 most common words account for 48.12% of all English texts. Let $c(w)$ be the number of taps a user needs to do before the word suggested as most likely is the desired word w. Similar to Section 5.2, this leads to the following model for a power user who uses word completion without visual search for the 100 most frequent words (first summand) and types words in full otherwise (second summand, using $search_{GN}$ as the search is only among the completed words, which come first): $T_{GC}(w) = 48.12\%(c(w) \times tap + space) + 51.88\%(|w| \times tap + search_{GN} + space)$.

According to our simulation of the word-frequency based suggestion algorithm used in GC and MC, which is based on the 7,582 most frequent English words, the expected number of taps a user needs to make before one of the 100 most frequent words becomes the most likely suggestion is $c = 2.05$. This is lower than one might think, as the three most frequent words (the, of, and), which account for more than 14% of English texts, all use different fingers on their first tap, so each appears immediately as a most likely suggestion. Furthermore, our simulation reveals that six of the 100 most frequent words (my, or, if, now, our, then, go) are never shown as most likely suggestion; they typically make up 1.15% of English texts, therefore we shift this percentage from the first to the second summand in our model. As in Section 5.2, we substitute c, the average word length in English texts $|w| = 4.79$, and estimates of $search_{GN}$, tap and $space$. In order to related the model to QWERTY typing speed, we describe tap and $space$ as linear estimates of $type_K$. The result is $T_{GC} = 5.5 \times type_K + 78$; the corresponding W_{GC} can be approximated for typical QWERTY typing speeds (up to 80 wpm) with a linear lower bound of $WPM_{GC} = 0.95 \times wpm_K$ (compared to $0.76 \times wpm_K$ for novice users). By learning tap prefixes for frequent words so that these words can be selected quickly without visual search, TapGazer is expected to allow power users to achieve typing speed close to QWERTY typing. Even if a user learns tap prefixes only for the 10 most common words, this accounts for about 25.13% of English texts and the estimated speed is $WPM_{GC} = 0.82 \times wpm_K$. Power users can use the same approach for TapGazer without gaze tracking (MC), with estimates $WPM_{MC} = 0.78 \times wpm_K$ when learning prefixes for the 100 most frequent words and $WPM_{MC} = 0.65 \times wpm_K$ for the 10 most frequent words (compared to $0.54 \times wpm_K$ for novice users). When using

gaze tracking, if a power user furthermore learns where a frequent word appears for the first time in the suggestions, e.g. "after tapping the left ring finger 'with' appears at the centre left", then the power user could potentially look at the right suggestion and select it immediately, reducing $c = 1.28$ and leading to an estimate of $WPM_{GC} = 1.03 \times wpm_K$ for the 100 and $WPM_{GC} = 0.84 \times wpm_K$ for the 10 most frequent words. If a power user is willing to learn a new layout, i.e. a finger-to-letter mapping not based on QWERTY, then c can be reduced further. We used branch-and-bound search to find a mapping that minimises c for the 100 frequent words, resulting in a mapping with $c = 1.18$ and $WPM_{GC} = 1.05 \times wpm_K$ for learned prefixes only, if the positions of the respective word suggestions are also learned. In summary, learning tap prefixes and even display positions for common words can potentially speed TapGazer up drastically, with and without gaze tracking.

5.4 Discussion

Similar to KLM [12], our models are based on the average measurements obtained from the evaluation. As a result, their predictions will be inaccurate to to some degree when applied to different groups of users. In particular, our experiments collected TapGazer performance data only from novice users. A multi-level regression analysis shows that the effect of session number on wpm was significant ($B = 1.99, 95\%CI = [1.50, 2.47], t(398) = 8.09, p < .001^{***}$), indicating that participants increased their wpm by about 2 wpm for each usage session. It is likely that users will continue to get faster with practice. The models we created based on short term use are therefore likely to underestimate the performance of longer-term users, forming a reference baseline for future research. Also, the models add value by formalising strategies that some users will likely apply to increase their TapGazer performance. Finally, the models identify important parameters affecting TapGazer's performance, providing starting points for further improvements in future work.

6 CONCLUSION AND FUTURE WORK

We have presented TapGazer, a novel text entry method combining tapping and gaze. TapGazer was designed to facilitate text entry while wearing VR/AR HMDs, without the need for a physical keyboard or to look at one's hands, in anticipation of a future where affordable AR glasses will be as ubiquitous as mobile phones are today. Our results indicate that novice users can achieve 77% of their QWERTY typing speed, with an average TapGazer WPM of 52.17, which surpasses the performance of comparable text entry methods. Furthermore, the error rate of TapGazer is significantly lower than for a physical QWERTY keyboard. We have created performance models for TapGazer that illustrate how different users can benefit from different usage strategies, and identify important performance parameters that can be optimised in future design iterations. In future work, we anticipate longer-term studies of performance, the use of Tapgazer with different devices and in different poses, improvements to word prediction based on more sophisticated language models, and error correction.

REFERENCES

[1] Jiban Adhikary. 2018. Text Entry in VR and Introducing speech and gestures in VR text entry. In *MobileHCI*. Association for Computing Machinery, Barcelona, Spain, 1083–1092.

[2] Sunggeun Ahn and Geehyuk Lee. 2019. Gaze-Assisted Typing for Smart Glasses. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 857–869.

[3] Aaron Bangor, Philip T Kortum, and James T Miller. 2008. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction* 24, 6 (2008), 574–594.

[4] Nikolaus Bee and Elisabeth André. 2008. Writing with your eye: A dwell time free writing system adapted to the nature of human eye gaze. In *International Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems*. Springer, 111–122.

[5] Burak Benligiray, Cihan Topal, and Cuneyt Akinlar. 2019. SliceType: fast gaze typing with a merging keyboard. *Journal on Multimodal User Interfaces* 13, 4 (2019), 321–334.

[6] Xiaojun Bi, Barton A Smith, and Shumin Zhai. 2010. Quasi-qwerty soft keyboard optimization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 283–286.

[7] Gaddi Blumrosen, Katsuyuki Sakuma, John Jeremy Rice, and John Knickerbocker. 2020. Back to Finger-Writing: Fingertip Writing Technology Based on Pressure Sensing. *IEEE Access* 8 (2020), 35455–35468.

[8] Costas Boletsis and Stian Kongsvik. 2019. Controller-based Text-input Techniques for Virtual Reality: An Empirical Comparison. *International Journal of Virtual Reality* 19, 3 (2019), 2–15.

[9] Sidney Bovet, Aidan Kehoe, Katie Crowley, Noirin Curran, Mario Gutierrez, Mathieu Meisser, Damien O Sullivan, and Thomas Rouvinez. 2018. Using traditional keyboards in VR: SteamVR developer kit and pilot game user study. In *2018 IEEE Games, Entertainment, Media Conference (GEM)*. IEEE, 1–9.

[10] Doug Bowman, Vinh Ly, Joshua Campbell, and Virginia Tech. 2001. Pinch keyboard: Natural text input for immersive virtual environments. (01 2001).

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

[12] Stuart K Card, Thomas P Moran, and Allen Newell. 1980. The keystroke-level model for user performance time with interactive systems. *Commun. ACM* 23, 7 (1980), 396–410.

[13] Steven J Castellucci, I Scott MacKenzie, Mudit Misra, Laxmi Pandey, and Ahmed Sabbir Arif. 2019. TiltWriter: design and evaluation of a no-touch tilt-based text entry method for handheld devices. In *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*. 1–8.

[14] Morokot Cheat and Manop Wongsaisuwan. 2018. Eye-Swipe Typing Using Integration of Dwell-Time and Dwell-Free Method. In *2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. IEEE, 205–208.

[15] Sibo Chen, Junce Wang, Santiago Guerra, Neha Mittal, and Soravis Prakkamakul. 2019. Exploring Word-gesture Text Entry Techniques in Virtual Reality. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6.

[16] Jacob Cohen. 1992. A power primer. *Psychological Bulletin* 112, 1 (1992), 155.

[17] Gennaro Costagliola, Vittorio Fuccella, and Michele Di Capua. 2011. Text entry with keyscretch. In *Proceedings of the 16th international conference on Intelligent user interfaces*. 277–286.

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

[19] Tafadzwa Joseph Dube and Ahmed Sabbir Arif. 2020. Impact of Key Shape and Dimension on Text Entry in Virtual Reality. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–10.

[20] John Dudley, Hrvoje Benko, Daniel Wigdor, and Per Ola Kristensson. 2019. Performance envelopes of virtual keyboard text input strategies in virtual reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 289–300.

[21] John J Dudley, Keith Vertanen, and Per Ola Kristensson. 2018. Fast and precise touch-based text entry for head-mounted augmented reality with variable occlusion. *ACM Transactions on Computer-Human Interaction (TOCHI)* 25, 6 (2018), 1–40.

[22] Mark D Dunlop, Naveen Durga, Sunil Motaparti, Prima Dona, and Varun Medapuram. 2012. QWERTH: an optimized semi-ambiguous keyboard design. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services companion*. 23–28.

[23] Francine Evans, Steven Skiena, and Amitabh Varshney. 1999. VType: Entering text in a virtual world. *submitted to International Journal of Human-Computer Studies* (1999).

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

[25] John M Findlay. 1997. Saccade target selection during visual search. *Vision research* 37, 5 (1997), 617–631.

[26] Yulia Gizatdinova, Oleg Špakov, and Veikko Surakka. 2012. Comparison of video-based pointing and selection techniques for hands-free text entry. In *Proceedings of the international working conference on advanced visual interfaces*. 132–139.

[27] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. Wristext: One-handed text entry on smartwatch using wrist gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–14.

[28] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language modeling for soft keyboards. In *Proceedings of the 7th international conference on Intelligent user interfaces*. 194–195.

[29] Nathan Green, Jan Kruger, Chirag Faldu, and Robert St. Amant. 2004. A reduced QWERTY keyboard for mobile text entry. In *CHI'04 extended abstracts on Human factors in computing systems*. 1429–1432.

[30] Jens Grubert, Lukas Witzani, Eyal Ofek, Michel Pahud, Matthias Kranz, and Per Ola Kristensson. 2018. Text entry in immersive head-mounted display-based virtual reality using standard keyboards. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 159–166.

[31] Jan Gugenheimer, David Dobbelstein, Christian Winkler, Gabriel Haas, and Enrico Rukzio. 2016. Facetouch: Enabling touch interaction in display fixed uis for mobile virtual reality. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 49–60.

[32] Aakar Gupta, Cheng Ji, Hui-Shyong Yeo, Aaron Quigley, and Daniel Vogel. 2019. RotoSwype: Word-Gesture Typing Using a Ring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[33] Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D Twigg, and Kenrick Kin. 2018. Online optical marker-based hand tracking with deep labels. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10.

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

[35] Anke Huckauf and Mario H Urbina. 2008. Gazing with pEYEs: towards a universal input for various applications. In *Proceedings of the 2008 symposium on Eye tracking research & applications*. 51–54.

[36] Robert JK Jacob. 1993. Eye movement-based human-computer interaction techniques: Toward non-command interfaces. *Advances in human-computer interaction* 4 (1993), 151–190.

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

[38] Haiyan Jiang, Dongdong Weng, Zhenliang Zhang, Yihua Bao, Yufei Jia, and Mengman Nie. 2018. HiKeyb: High-Efficiency Mixed Reality System for Text Entry. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, 132–137.

[39] Haiyan Jiang, Dongdong Weng, Zhenliang Zhang, and Feng Chen. 2019. HiFinger: One-Handed Text Entry Technique for Virtual Environments Based on Touches between Fingers. *Sensors* 19, 14 (2019), 3063.

[40] Sunjun Kim, Jeongmin Son, Geehyuk Lee, Hwan Kim, and Woohun Lee. 2013. TapBoard: making a touch screen keyboard more touchable. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 553–562.

[41] Pascal Knierim, Valentin Schwind, Anna Maria Feit, Florian Nieuwenhuizen, and Niels Henze. 2018. 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*. 1–9.

[42] Per-Ola Kristensson and Shumin Zhai. 2004. SHARK2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 43–52.

[43] Falko Kuester, Michelle Chen, Mark E Phair, and Carsten Mehring. 2005. Towards keyboard independent touch typing in VR. In *Proceedings of the ACM symposium on Virtual reality software and technology*. 86–95.

[44] Chandan Kumar, Ramin Hedeshy, Scott MacKenzie, and Steffen Staab. 2020. TAGSwipe: Touch Assisted Gaze Swipe for Text Entry. (2020).

[45] Manu Kumar, Andreas Paepcke, and Terry Winograd. 2007. EyePoint: practical pointing and selection using gaze and keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 421–430.

[46] Andrew Kurauchi, Wenxin Feng, Ajjen Joshi, Carlos Morimoto, and Margrit Betke. 2016. EyeSwipe: Dwell-free text entry using gaze paths. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1952–1956.

[47] Andrew Toshiaki Nakayama Kurauchi. 2018. *EyeSwipe: text entry using gaze paths.* Ph.D. Dissertation. Universidade de São Paulo.

[48] Lik Hang Lee, Kit Yung Lam, Tong Li, Tristan Braud, Xiang Su, and Pan Hui. 2019. Quadmetric Optimized Thumb-to-Finger Interaction for Force Assisted One-Handed Text Entry on Mobile Headsets. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (2019), 1–27.

[49] Minkyung Lee, Woontack Woo, et al. 2003. ARKB: 3D vision-based Augmented Reality Keyboard. In *ICAT*.

[50] Seongil Lee, Sang Hyuk Hong, and Jae Wook Jeon. 2002. Designing a universal keyboard using chording gloves. *ACM SIGCAPH computers and the physically handicapped* 73-74 (2002), 142–147.

[51] Frank Chun Yat Li, Richard T Guy, Koji Yatani, and Khai N Truong. 2011. The 1line keyboard: a QWERTY layout in a single line. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 461–470.

[52] Jia-Wei Lin, Ping-Hsuan Han, Jiun-Yu Lee, Yang-Sheng Chen, Ting-Wei Chang, Kuan-Wen Chen, and Yi-Ping Hung. 2017. Visualizing the keyboard in virtual reality for enhancing immersive experience. In *ACM SIGGRAPH 2017 Posters*. 1–2.

[53] Yi Liu, Chi Zhang, Chonho Lee, Bu-Sung Lee, and Alex Qiang Chen. 2015. Gazetry: Swipe text typing using gaze. In *Proceedings of the annual meeting of the australian special interest group for computer human interaction*. 192–196.

[54] Xueshi Lu, Difeng Yu, Hai-Ning Liang, Xiyu Feng, and Wenge Xu. 2019. DepthText: Leveraging Head Movements towards the Depth Dimension for Hands-free Text Entry in Mobile Virtual Reality Systems. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 1060–1061.

[55] Yiqin Lu, Chun Yu, Xin Yi, Yuanchun Shi, and Shengdong Zhao. 2017. Blindtype: Eyes-free text entry on handheld touchpad by leveraging thumb's muscle memory. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 1–24.

[56] Christof Lutteroth, Moiz Penkar, and Gerald Weber. 2015. Gaze vs. mouse: A fast and accurate gaze-only click alternative. In *Proceedings of the 28th annual ACM symposium on user interface software & technology*. 385–394.

[57] Xinyao Ma, Zhaolin Yao, Yijun Wang, Weihua Pei, and Hongda Chen. 2018. Combining brain-computer interface and eye tracking for high-speed text entry in virtual reality. In *23rd International Conference on Intelligent User Interfaces*. 263–267.

[58] I Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, and Eugene Skepner. 2001. LetterWise: Prefix-based disambiguation for mobile text input. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*. 111–120.

[59] I Scott MacKenzie and R William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *CHI'03 extended abstracts on Human factors in computing systems*. 754–755.

[60] I Scott MacKenzie and Shawn X Zhang. 1999. The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 25–31.

[61] I Scott MacKenzie and Xuang Zhang. 2008. Eye typing using word and letter prediction and a fixation algorithm. In *Proceedings of the 2008 symposium on Eye tracking research & applications*. 55–58.

[62] Päivi Majaranta, Ulla-Kaija Ahola, and Oleg Špakov. 2009. Fast gaze typing with an adjustable dwell time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 357–360.

[63] Päivi Majaranta and Kari-Jouko Räihä. 2007. Text entry by gaze: Utilizing eye-tracking. *Text entry systems: Mobility, accessibility, universality* (2007), 175–187.

[64] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: a mid-air word-gesture keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1073–1082.

[65] Brian McElree and Marisa Carrasco. 1999. The temporal dynamics of visual search: evidence for parallel processing in feature and conjunction searches. *Journal of Experimental Psychology: Human Perception and Performance* 25, 6 (1999), 1517.

[66] Mark McGill, Daniel Boland, Roderick Murray-Smith, and Stephen Brewster. 2015. 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*. 2143–2152.

[67] Martez E Mott, Shane Williams, Jacob O Wobbrock, and Meredith Ringel Morris. 2017. Improving dwell-based gaze typing with dynamic, cascading dwell times. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2558–2570.

[68] Peter Norvig. 2013. English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU. (2013).

[69] Midori Ohkita, Yoshie Obayashi, and Masako Jitsumori. 2014. Efficient visual search for multiple targets among categorical distractors: Effects of distractor–distractor similarity across trials. *Vision research* 96 (2014), 96–105. https://doi.org/10.1016/j.visres.2014.01.009

[70] Jakob Olofsson. 2017. Input and Display of Text for Virtual Reality Head-Mounted Displays and Hand-held Positionally Tracked Controllers.

[71] Alexander Otte, Tim Menzner, Travis Gesslein, Philipp Gagel, Daniel Schneider, and Jens Grubert. 2019. Towards utilizing touch-sensitive physical keyboards for text entry in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 1729–1732.

[72] Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen, and Per Ola Kristensson. 2013. Improving two-thumb text entry on touchscreen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2765–2774.

[73] Farshid Salemi Parizi, Eric Whitmire, and Shwetak Patel. 2019. AuraRing: Precise Electromagnetic Finger Tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–28.

[74] Diogo Pedrosa, Maria Da Graça Pimentel, Amy Wright, and Khai N Truong. 2015. Filteryedping: Design challenges and user performance of dwell-free eye typing. *ACM Transactions on Accessible Computing (TACCESS)* 6, 1 (2015), 1–37.

[75] Abdul Moiz Penkar, Christof Lutteroth, and Gerald Weber. 2012. Designing for the eye: design parameters for dwell in gaze interaction. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*. 479–488.

[76] Ken Perlin. 1998. Quikwriting: continuous stylus-based text entry. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*. 215–216.

[77] Duc-Minh Pham and Wolfgang Stuerzlinger. 2019. HawKEY: Efficient and Versatile Text Entry for Virtual Reality. In *25th ACM Symposium on Virtual Reality Software and Technology*. 1–11.

[78] Ryan Qin, Suwen Zhu, Yu-Hao Lin, Yu-Jung Ko, and Xiaojun Bi. 2018. Optimal-t9: An optimized t9-like keyboard for small touchscreen devices. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. 137–146.

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

[80] Jochen Rick. 2010. Performance optimizations of virtual keyboards for stroke-based text entry on a touch-based tabletop. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. 77–86.

[81] Sherry Ruan, Jacob O Wobbrock, Kenny Liou, Andrew Ng, and James A Landay. 2018. Comparing speech and keyboard text entry for short messages in two languages on touchscreen phones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–23.

[82] Sayan Sarcar, Prateek Panwar, and Tuhin Chakraborty. 2013. EyeK: an efficient dwell-free eye gaze-based text entry system. In *Proceedings of the 11th asia pacific conference on computer human interaction*. 215–220.

[83] Weinan Shi, Chun Yu, Xin Yi, Zhen Li, and Yuanchun Shi. 2018. TOAST: Ten-finger eyes-free typing on touchable surfaces. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–23.

[84] Shyamli Sindhwani, Christof Lutteroth, and Gerald Weber. 2019. ReType: Quick Text Editing with Keyboard and Gaze. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[85] Brian A Smith, Xiaojun Bi, and Shumin Zhai. 2015. Optimizing touchscreen keyboards for gesture typing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3365–3374.

[86] R William Soukoreff and I Scott MacKenzie. 2003. Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 113–120.

[87] Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. LuminosoInsight/wordfreq: v2.2. https://doi.org/10.5281/zenodo.1443582

[88] Marco Speicher, Anna Maria Feit, Pascal Ziegler, and Antonio Krüger. 2018. Selection-based text entry in virtual reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[89] Srinath Sridhar, Anna Maria Feit, Christian Theobalt, and Antti Oulasvirta. 2015. 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*. 3643–3652.

[90] FirstName Surname. 2018. Text Input in Virtual Reality Using a Tracked Drawing Tablet. (2018).

[91] Bruce H Thomas and Wayne Piekarski. 2002. Glove based user interaction techniques for augmented reality in an outdoor environment. *Virtual Reality* 6, 3 (2002), 167–180.

[92] Keith Vertanen, Crystal Fletcher, Dylan Gaines, Jacob Gould, and Per Ola Kristensson. 2018. The impact of word, multiple word, and sentence input on virtual keyboard decoding performance. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[93] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 659–668.

[94] James Walker, Bochao Li, Keith Vertanen, and Scott Kuhl. 2017. Efficient typing on a visually occluded physical keyboard. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 5457–5461.

[95] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. 14–27.

[96] David J Ward, Alan F Blackwell, and David JC MacKay. 2000. Dasher—a data entry interface using continuous gestures and language models. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*. 129–137.

[97] Eric Whitmire, Mohit Jain, Divye Jain, Greg Nelson, Ravi Karkar, Shwetak Patel, and Mayank Goel. 2017. Digitouch: Reconfigurable thumb-to-finger input and text entry on head-mounted displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–21.

[98] Pui Chung Wong, Kening Zhu, and Hongbo Fu. 2018. Fingert9: Leveraging thumb-to-finger interaction for same-side-hand text entry on smartwatches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–10.

[99] Wenge Xu, Hai-Ning Liang, Anqi He, and Zifan Wang. 2019. Pointing and Selection Methods for Text Entry in Augmented Reality Head Mounted Displays. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 279–288.

[100] Wenge Xu, Hai-Ning Liang, Yuxuan Zhao, Tianyu Zhang, Difeng Yu, and Diego Monteiro. 2019. 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 (2019), 1991–2001.

[101] Zheer Xu, Weihao Chen, Dongyang Zhao, Jiehui Luo, Te-Yen Wu, Jun Gong, Sicheng Yin, Jialun Zhai, and Xing-Dong Yang. 2020. BiTipText: Bimanual Eyes-Free Text Entry on a Fingertip Keyboard. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[102] Naoki Yanagihara, Buntarou Shizuki, and Shin Takahashi. 2019. Text Entry Method for Immersive Virtual Environments Using Curved Keyboard. In *25th ACM Symposium on Virtual Reality Software and Technology*. 1–2.

[103] Xin Yi, Chen Wang, Xiaojun Bi, and Yuanchun Shi. 2020. PalmBoard: Leveraging Implicit Touch Pressure in Statistical Decoding for Indirect Text Entry. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[104] Xin Yi, Chun Yu, Mingrui Zhang, Sida Gao, Ke Sun, and Yuanchun Shi. 2015. 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 & Technology*. 539–548.

[105] Yafeng Yin, Qun Li, Lei Xie, Shanhe Yi, Edmund Novak, and Sanglu Lu. 2016. CamK: A camera-based keyboard for small mobile devices. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

[106] Chun Yu, Yizheng Gu, Zhican Yang, Xin Yi, Hengliang Luo, and Yuanchun Shi. 2017. Tap, dwell or gesture? Exploring head-based text entry techniques for HMDs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 4479–4488.

[107] Chun Yu, Ke Sun, Mingyuan Zhong, Xincheng Li, Peijun Zhao, and Yuanchun Shi. 2016. One-dimensional handwriting: Inputting letters and words on smart glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 71–82.

[108] Difeng Yu, Kaixuan Fan, Heng Zhang, Diego Monteiro, Wenge Xu, and Hai-Ning Liang. 2018. PizzaText: text entry for virtual reality systems using dual thumbsticks. *IEEE Transactions on Visualization and Computer Graphics* 24, 11 (2018), 2927–2935.

[109] Shumin Zhai, Michael Hunter, and Barton A Smith. 2000. The metropolis keyboard-an exploration of quantitative techniques for virtual keyboard design. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*. 119–128.

[110] Shumin Zhai, Michael Hunter, and Barton A Smith. 2002. Performance optimization of virtual keyboards. *Human–Computer Interaction* 17, 2-3 (2002), 229–269.

[111] Mingrui Ray Zhang, He Wen, and Jacob O Wobbrock. 2019. Type, Then Correct: Intelligent Text Correction Techniques for Mobile Text Entry Using Neural Networks. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 843–855.

[112] Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an invisible keyboard. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.