WinkSwype: A Wink-Based, Hands-Free Shape-Writing Method for VR Using Single-Eye Calibration

Tafadzwa Joseph Dube
Inclusive Interaction Lab
University of California, Merced
Merced, California, United States
tdube@ucmerced.edu

Qhelile Ozias Sibanda
Inclusive Interaction Lab
University of California, Merced
Merced, California, United States
qsibanda@ucmerced.edu

I. Scott MacKenzie
Electrical Engineering and Computer Science
York University
Toronto, Ontario, Canada
mack@yorku.ca

Ahmed Sabbir Arif
Inclusive Interaction Lab
University of California, Merced
Merced, California, United States
asarif@ucmerced.edu

(a) (b) (c)

Figure 1: (a) The testing application showing the presented text, transcribed text, suggestion bar, and virtual Qwerty keyboard. The red dot is the current gaze-controlled cursor position. (b) A participant entering text using a controller-based method. The base station for headset tracking is on a box on the table. (c) A participant entering text with WinkSwype using only gaze input and not holding a controller.

Abstract

Text entry in virtual reality (VR) remains challenging due to the limited field of view and the inability to see and reliably locate external physical devices. Existing methods often rely on external hardware or sustained physical actions, limiting accessibility. We present WinkSwype, a novel hands-free, wink-based shape-writing technique where users trace the letters of a word using single-eye gaze. To support this, we developed a single-eye calibration model and a hybrid gesture recognition algorithm optimized for single-eye gaze input. We evaluated WinkSwype against two established techniques: controller-based raycasting and gaze-assisted controller input. Participants retained 58-71% of their text entry speed with WinkSwype while achieving comparable error rates and usability ratings. Subjective feedback indicated no significant increase in workload or visual fatigue during short-term use. These findings suggest its potential as a hands-free text entry method for VR.

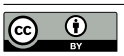
CCS Concepts

• **Human-centered computing** → **Virtual reality**; **Text input**; *Gestural input*; Pointing; *Interaction design.*

Keywords

Text entry, Controllers, Gaze pointing, Eye tracking, Virtual reality, Gesture typing, Shape-writing

ACM Reference Format:
Tafadzwa Joseph Dube, Qhelile Ozias Sibanda, I. Scott MacKenzie, and Ahmed Sabbir Arif. 2026. WinkSwype: A Wink-Based, Hands-Free Shape-Writing Method for VR Using Single-Eye Calibration. In *Proceedings of The 52nd Graphics Interface Conference (GI '26).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX



GI '26, Kitchener-Waterloo, Ontario, Canada

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Text entry in VR remains a persistent challenge due to several constraints: the limited field of view (FOV) in head-mounted displays, the unavailability of users' hands, and the inability to see external

physical devices. A variety of input methods have been explored, including physical Qwerty keyboards, virtual keyboards, controllers, and wearable devices (§2). However, many of these break immersion, rely on external hardware, or require physical actions that are not accessible to individuals with motor impairments and in dual-task situations where hands are otherwise engaged [10].

Gaze-based interaction and gesture typing offer promising alternatives, but each comes with trade-offs in speed, accuracy, and usability. Physical Qwerty keyboards deliver the fastest input speeds, but typically require external tracking systems that compromise accessibility and immersion (§2). Hands-free methods such as head pointing and mid-air typing address some of these concerns but often suffer from low performance, user fatigue, and high cognitive load. Despite growing interest, accessibility remains an underexplored aspect of VR text input, with many systems failing to support users with diverse physical needs [10].

Gesture typing, or shape-writing, is a word-level predictive input method in which users trace a path across the keys of a virtual keyboard, connecting the letters of the intended word. The system then predicts the most probable word based on the shape of the gesture. In VR, this technique uses either gaze or handheld controllers [43]. However, a key challenge is gesture framing, that is, determining when a gesture begins and ends. Most approaches rely on continuous input (e.g., holding a button) or separate initiation and completion actions, which can reduce both usability and accessibility, especially in hands-free scenarios.

This work introduces WinkSwype, a novel, fully hands-free shape-writing method that leverages gaze tracking for text entry in VR. Users initiate a gesture by closing one eye (a "wink") and complete the gesture by reopening the eye, eliminating the need for any additional physical input. The design was iterative, with each design decision informed by multiple pilot studies. Since current commercial eye trackers do not support single-eye tracking natively, we first developed a custom single-eye calibration model to improve gaze precision when one eye is closed. We then enhanced existing gesture recognition algorithms to better support gaze-based shape-writing. Finally, we evaluated WinkSwype against two common VR text input methods, each adapted to use our optimized recognition pipeline: a controller-based raycasting method, in which users trace words while holding the trigger, and a gaze-trigger method, where users gaze at letters while pressing the controller trigger to frame gestures.

2 Related Work

Recent work has investigated a variety of text input techniques in VR [10]. The fastest methods rely on physical Qwerty keyboards, reaching speeds of up to 39 words per minute (wpm) by using external sensors to track hands and render them in VR [4, 14–16, 21, 22]. In contrast, mid-air virtual keyboards eliminate the need for physical hardware but achieve lower speeds, around 12 wpm [8, 12, 39]. Other approaches include handheld [6, 18, 39] and wearable devices like thimbles [11]. While promising, these techniques can be physically fatiguing, rely on additional tracking hardware, or may not support users with motor impairments or those experiencing situational limitations where hands are unavailable.

2.1 Hands-free Text Entry

Several hands-free text input techniques have been proposed to support users with motor impairments, enable multitasking, or accommodate temporary situations where hands are unavailable. One common approach is head pointing, where users control a cursor on a virtual keyboard and select characters via dwell. Reported speeds include 10.59 wpm with a 3.7% error rate [45], and lower performance on alternative layouts such as circular (6.03 wpm) and Qwerty (8.33 wpm) keyboards, with error rates ranging from 6-8% [44]. Dwell-free variants like RingText improved performance, achieving 13.24 wpm and a 2.9% error rate for expert users [44], while swipe-based methods adapted for head pointing reached around 6.3 wpm [29].

Some techniques separate pointing and selection using blinks or neck movements [32]. For example, Lu et al. [29] found that blink-based selection yielded the best performance among three hands-free methods, with a text entry speed of about 13 wpm. However, these methods often suffer from fatigue and false activations. Speech activation has also been explored, achieving around 13 wpm, though it struggles in noisy environments [5]. Lip-based input methods, such as LipText, offer an alternative, with novices reaching 8.63 wpm and experts up to 9.8 wpm, though with error rates between 5-7% depending on user experience [26].

Table 1: Comparison of gaze-based text entry methods in VR. Columns report selection method, input granularity, text entry speed (wpm), and error rate (or † word error rate). * indicates systems using non-Qwerty keyboard layouts. The final row corresponds to the proposed method.

Reference	Selection	Input Level	Speed	Error Rate
Sarcar et al. [37]*	Gaze Area	Character-based	6.0	16.90%
Rajanna and Hansen [36]	Dwell	Character-based	9.4	0.02%
Rajanna and Hansen [36]	Button Press	Character-based	10.2	0.07%
Cui et al. [9]	Gaze Area	Word-based	10.9	2.71%†
Zhao et al. [46]	Hand Gesture (IMU)	Word-based	17.6	1.01%
Mutasim et al. [33]	Multi-threshold Dwell	Character-based	17.8	2.00%
Ma et al. [31]*	Gaze & EEG	Character-based	10.0	–
WinkSwype	Wink	Word-based	4.88	2.23%

2.2 Gaze-based Text Entry

Gaze typing enables users to input text by directing their eye movements over a virtual keyboard, making it especially suitable for individuals with limited motor abilities [30]. These methods are broadly categorized into dwell-based and dwell-free approaches [46]. Dwell-based techniques rely on users fixating on a key for a set duration, typically between 400 and 1500 milliseconds. Most gaze-based text input systems for VR use this approach. For example, Rajanna and Hansen [36] compared dwell-based selection (550 ms) with external button presses, finding that button input performed better (10.2 wpm vs. 9.4 wpm). Mutasim et al. [33] evaluated three dwell-based techniques: a standard constant-threshold method (450 ms), a dual-threshold method (300/500 ms), and a multi-threshold keyboard. The multi-threshold approach achieved the highest speed (17.8 wpm), outperforming the constant-threshold (12.7 wpm) and dual-threshold (15.2 wpm) systems.

To address the inherent slowness of dwell, dwell-free methods have been proposed. These approaches often use gaze gestures or auxiliary input devices to signal selection. For instance, Sarcar et al. [37] introduced a technique in which users glance at a key, look away, and then return to it to confirm selection. Similarly, Cui et al. [9] employed a glance-away approach to frame gestures in gaze-based shape-writing. Other systems use predefined gaze strokes (e.g., EyeWrite), or rely on gaze-based command triggers such as looking at the Space key to finalize words (e.g., GazeTry [28]). Multimodal systems have also emerged. GazeSpeedup combines gaze with data from a wrist-worn IMU, achieving 17.6 wpm [46]. Similarly, Ma et al. [31] integrated gaze with a brain-computer interface (BCI), reaching 10 wpm. Other hybrid approaches include gaze paired with finger-pointing (10.7 wpm [30]) and gaze combined with hand movements (9.1 wpm) [36]. While these methods show promise, many require external devices or secondary motor actions for confirmation, limiting their accessibility and scalability. Moreover, some have not been rigorously evaluated. Word-level gaze input methods have also been proposed, though gesture framing remains a core challenge. For example, EyeSwipe uses a reverse cross gesture to delimit word boundaries, reaching 11.7 wpm after brief training [25]. Other techniques require users to look at specific buttons to mark the start and end of gestures [17]. TAGSwipe, a multimodal method, combines gaze with a single press-and-release touch gesture to signal word boundaries, achieving 15.5 wpm [24]. However, such methods still rely on physical actions to frame gestures, which may limit accessibility for hands-free use. Table 1 presents a comparison of gaze-based text entry methods in VR.

3 WinkSwype

WinkSwype enables users to shape-write in VR using gaze pointing. Users control the cursor with their gaze and use a wink to frame the gesture: they close one eye over the first letter of the intended word, gaze over the intermediate letters, and open the eye on the last letter to complete the gesture. A predictive system then automatically enters the most probable word along with a space character. Users can replace the entered word by dwelling on a suggestion in the suggestion bar for 1,000 milliseconds. The suggestion bar contains the 20 most probable words but displays four at a time. Users can cycle through suggestions by dwelling on the left or right arrow keys. Dwelling on the backspace key deletes the entire word from the input field.

We initially explored an alternative interaction in which users wink at the first letter and wink again at the last letter to frame the gesture. However, pilot trials showed that this approach increased input time and introduced additional errors due to the Heisenberg effect (i.e., interaction with the system interfering with input precision [3]). We therefore discarded this approach.

The system also supports character-level text entry and editing actions (e.g., shift, switching to number keys) by dwelling on the corresponding virtual keys. The system provides visual feedback throughout the input process. Initially, the cursor appears red, but turns green during shape writing. In addition, a circular progress indicator informs users of the remaining dwell time required to select a target. Fig. 1a shows the keyboard layout, measuring 1.1×0.79 meters, with a total visual angle of 16.5° and approximately 1.5° per key. It was developed using Unity 2019.2.0f1, with integration of SteamVR, VIVE Eye Tracking SDK 1.3.6.8, and SRanipal SDK 1.0.1.0 to capture and interpret eye-tracking data.

4 Modeling Single-Eye Calibration

Preliminary investigations revealed that standard dual-eye calibration was insufficient to correct the gaze offset that occurred when users closed one eye, primarily due to the change in visual perspective. This issue was further amplified by the use of a relatively smaller keyboard compared to those used in existing gaze-based shape-writing systems. We initially attempted to calibrate the tracker with one eye closed, intending to switch between calibration parameters for both-eyes-open and one-eye-closed scenarios. However, the system remained inaccurate even with one-eye-closed calibration, due to inherent limitations in the underlying calibration algorithm.

To address this issue, we developed a custom single-eye calibration procedure, performed with one eye closed, to compensate for the perspective shift and improve gaze precision. This method was refined iteratively through trial and evaluation, with adjustments based on observed inaccuracies. By combining standard calibration for both-eyes-open scenarios with our custom calibration for one-eye-closed use, we significantly improved the precision of WinkSwype. The following sections describe our proposed single-eye calibration process.

4.1 Data Collection

The custom calibration procedure was tailored to the requirements of our virtual environment. We began by calculating the display size (width) as $W = 2 \cdot d \cdot \tan(fov/2)$, where d represents the distance from the user's eyes to the target, and $fov = 60^\circ$ to approximate the central visual field of the human eye, and a 3.8 m distance, determined through lab trials, to be ideal for our setup. This resulted in a display area of 4.5×4.5 m or 20.25 m^2.

To identify the area of lowest error, we segmented the display area into nine equal zones. Within each zone, 20 red sphere targets were presented sequentially (Fig. 2b). In a pilot study with two participants, each participant closed one eye to simulate single-eye interaction. They were then asked to focus on each target and press a button on the controller when they felt they were looking

directly at it. For each target, we recorded both the actual target position and the user's eye gaze position. Since our application used a flat keyboard, the z-coordinate was kept constant. We calculated the average correction vector for each zone. Zone 5 exhibited the smallest average correction vector, indicating minimal error. This led to our decision to place all interactive elements, including the keyboard and options, within Zone 5. We then investigated three distinct calibration methods specifically for the selected zone.

4.2 Ratio-Based Calibration

We first explored a ratio-based approach, in which the system adjusted user-reported gaze positions to match actual target positions using scaling factors. Specifically, the calibration factors for the x-axis was calculated as $C_x = T_x/R_x$ and y-axis was calculated as $C_y = T_y/R_y$, where T_x and T_y are the actual target positions and R_x and R_y are the raw gaze positions. We then computed the mean calibration factor for the x-axis as $\mu_x = (\sum_{i=1}^{n} C_{x_i})/n$, and for the y-axis as $\mu_y = (\sum_{i=1}^{n} C_{y_i})/n$ to improve robustness, where n is the number of calibration points. These average calibration factors were subsequently used to scale user-reported positions, minimizing positional errors.

4.3 Linear Regression-Based Calibration

We also explored a linear fitting approach to correct positional inaccuracies by modeling the relationship between recorded and target positions using linear regression for the x-axis $C_x = m_x \cdot R_x + b_x$ and the y-axis $C_y = m_y \cdot R_y + b_y$, where m and b are the slope and intercept coefficients fitted by linear regression. The calculated values were then used to map the user-reported positions to their corresponding calibrated positions.

4.4 Gaussian Error Modeling Calibration

Finally, we explored a Gaussian-based calibration method. This approach models the error between raw eye-tracking coordinates and their corresponding actual positions. Instead of fitting a Gaussian distribution directly to the raw data, we first computed the deviation (error) between each eye-tracking point and its associated ground-truth coordinate. These error values were then used to fit a Gaussian distribution along both the x-axis $\epsilon_{x_i} = R_{x_i} - T_{x_i}$, and y-axis $\epsilon_{x_i} = R_{x_i} - T_{x_i}$, where ϵ represents deviation or error, T_x and T_y are the actual target positions, and R_x and R_y are the raw gaze positions recorded, as before, and i represents each of the calibration data points. We then calculated the means for the x-axis as $\mu_{\epsilon_x} = 1/20 \sum_{i=1}^{n} \epsilon_{x_i}$ and y-axis as $\mu_{\epsilon_y} = 1/20 \sum_{i=1}^{n} \epsilon_{y_i}$, and the standard deviations (σ) for both axes using Equations 1 and 2, respectively.

$$\sigma_{\epsilon_x} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\epsilon_{x_i} - \mu_{\epsilon_x})^2} \tag{1}$$

$$\sigma_{\epsilon_y} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\epsilon_{y_i} - \mu_{\epsilon_y})^2} \tag{2}$$

These resulting Gaussian parameters, specifically the mean and standard deviation of the error distribution, were used to estimate and correct errors in new eye-tracking data. Finally, we obtained the calibrated x and y positions by subtracting the predicted error from the raw eye-tracking input: $C_x = \epsilon_x - \mu_{\epsilon_x}$ and $C_y = \epsilon_y - \mu_{\epsilon_y}$, respectively.

4.5 Comparing Calibration Approaches

We compared the three calibration approaches in a pilot study with two participants (1 male, 1 female; M = 31.5 years, SD = 2.5), both of whom had a dominant right eye. During the trial, each participant fixated on every letter of the virtual keyboard while keeping one eye closed. We then measured the precision of the calibrated gaze positions using each of the three methods.

The ratio-based calibration model yielded a precision of 1.32°, the linear regression model achieved 1.17°, and the Gaussian error modeling approach produced the best precision at 1.07° of visual angle. Given its superior performance, we adopted the Gaussian model for the final study. For context, the default eye-tracking precision of the Vive Pro Eye tracker ranges between $0.5 - 1.1°$ [7].

5 Gesture Recognition

To optimize gesture recognition, we first evaluated two existing word prediction methods, then based on the findings, developed a custom hybrid approach that combines the strengths of both.

5.1 String Matching-Based Recognition

The string matching-based approach, adapted from Kaufman [20], records the fixation time on all letters the user gazes over while shape-writing. It then applies the Dynamic Time Warping (DTW) algorithm to match the resulting sequence with words from a dictionary. DTW accommodates variations in timing and sequence length by computing the optimal alignment path between two sequences: the input sequence S_g and a dictionary word sequence S_w. To improve accuracy, the algorithm incorporates a weighting factor that assigns greater importance to letters with longer fixation durations. DTW computes a distance matrix D, where each element $D(i, j)$ is recursively defined as in Equation 3. Here, i and j denote the row and column indices in the matrix, and ω is a weighting factor derived from fixation time F_t using Equation 4.

$$D(i,j) = \min \begin{cases} D(i-1,j) + \omega, \\ D(i,j-1) + \omega, \\ D(i-1,j-1) + \omega \end{cases} \tag{3}$$

$$\omega = \begin{cases} 0, & \text{if } S_g[i] = S_w[j], \\ F_t[i], & \text{otherwise.} \end{cases} \tag{4}$$

For non-gaze-based methods, the alignment between the input sequence and the word template is computed as an unweighted optimal path, as temporal weighting (e.g., fixation duration) is not available. In our implementation, we used the 10,000 most common English words [20] and optimized the search process by constraining the first letter of the candidate words. This letter is determined based on the first character where the user either begins tracing with the controller or fixates for more than 150 milliseconds. This threshold was established through a pilot study that showed that users consistently fixed for more than 150 ms on the first letter of their intended word. Therefore, the final DTW distance, representing the dissimilarity between S_g and S_w, is given

by $DTW(S_g, S_w) = D(m, n)$, where m and n denote the lengths of the respective sequences:

5.2 Pattern Matching-Based Recognition

The pattern matching-based approach, developed by Dube et al. [11], Kristensson and Zhai [23], Yu et al. [45], compares the shape and spatial position of the user's gestures (either gaze paths or controller-drawn traces over the keyboard) to predefined gesture templates corresponding to dictionary words. To generate these templates, we programmatically created traces for the 10,000 most common English words [20] by connecting the centers of the corresponding letters in each word. Following prior recommendations, both the gestures and the word templates were resampled to 50 evenly spaced points to allow for consistent point-by-point comparison [11, 45]. To reduce the noise of saccades, unintentional gaze movements, and controller jitters, we applied Bresenham's line algorithm for gesture smoothing [40]. We also tested the Savitzky-Golay filter [42], but found it to be less effective in our lab tests.

As with the string matching-based approach, we first filtered candidate templates by identifying words that begin with the first character where the user either begins tracing with the controller or fixates for more than 150 milliseconds. We then performed a point-by-point comparison between the user's gesture and the filtered templates. The angular difference between the corresponding points was computed as $\Delta\theta = \sum_{i=1}^{n}(G\theta_i - T\theta_i)$. To estimate the spatial difference between the gesture and the template, we calculated the distance between each pair of corresponding points and calculated the total distance (D) between the shapes as $D = \sum_{i=1}^{n} \|G_i - T_i\|$.

5.3 Hybrid Rank-Based Recognition

We developed a hybrid approach that combines both string and pattern matching-based methods to enhance recognition performance. The method first identifies candidate words using the string matching-based approach, then ranks these candidates based on the similarity between the corresponding user gesture and the predefined template using the pattern matching-based approach. This design prioritizes the string matching-based method, as it consistently outperformed the pattern matching-based method in our lab trials. The hybrid approach operates in the following three steps:

(1) The combined word list from string and pattern matching is analyzed to determine the frequency (F) of each word, representing the number of times it appears in both methods.
(2) Then each word is assigned an order score (O) using the formula $O = \frac{i+1}{|W|\times 10}$, where i is the index of the word in the list and $|W|$ is the length of the word. This formula ensures that words appearing earlier in the list receive slightly higher (less negative) scores. Words that do not appear in the word list are assigned a fixed, lower order score (e.g., -0.4) to ensure they rank below equally frequent words from the string matching results.
(3) A total score (S) for each word is calculated by combining its frequency and order score: $S = F + O$. Words are then ranked in descending order according to their total score (S), with higher scores indicating greater relevance. The top-ranked word is then automatically entered into the input area, and the next 20 highest-ranked words are selected to display in the suggestion bar.

Table 2: Aggregate percentages for each ranking approaches based on target word position in the prediction list.

Recognition	1st	2nd	3rd	4th	Top 20%	Missing
String Matching	33.3%	33.3%	11.1%	11.1%	11.1%	0%
Pattern Matching	22.2%	11.1%	22.2%	11.1%	11.1%	22.2%
Hybrid Rank	66.7%	11.1%	0%	11.1%	11.1%	0%

5.4 Comparing Recognition Approaches

To compare the recognition approaches, we conducted a pilot study with two participants (both male; M = 30.5 years, SD = 4.5). Participants were asked to input two pangrams: *"a quick brown fox jumps over the lazy dog"* and *"the five boxing wizards jump quickly"*, using the system. These sentences ensured full coverage of the keyboard, enabling us to evaluate the effectiveness of the calibration process in supporting consistent and accurate gaze input across all letter positions. We recorded the percentage of times the correct word appeared as the top-ranked prediction for each of the three models. Table 2 presents the results, showing that the hybrid approach outperformed both the string and pattern matching-based approaches. Hence, we adopted this approach for our user study.

6 User Study: Evaluating WinkSwype

We conducted a within-subjects user study to evaluate the performance of WinkSwype compared to two alternative shape-writing methods, described below.

- **Raycasting** enables shape-writing using a ray projected from the handheld controller, following the approach described by Chen et al. [6]. The user aims the ray at the first letter of the target word, presses and holds the trigger of the same controller to trace the word across the virtual keyboard, then releases the trigger to finalize the input. This method enables character-based input and editing actions by aiming the ray at the desired key and pressing the trigger.
- **Gaze-Trigger** combines gaze pointing with controller input. Like WinkSwype, the user connects the letters of a target word using gaze pointing, but uses the controller trigger to frame the shape-writing gesture. The user first gazes at the initial letter of the target word, then presses and holds the trigger to begin gesturing, and releases the trigger to complete the input. This method enables character-based input and editing actions by fixating on the desired key and pressing the trigger to select it.

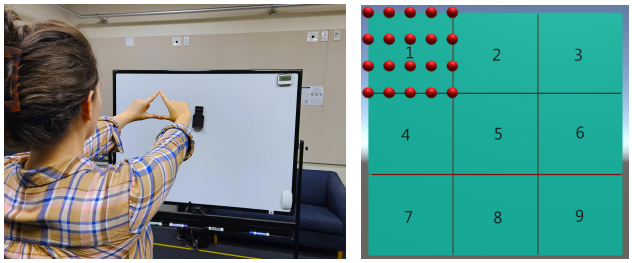
Both methods use the same user interface (Fig. 1a) and gesture recognition approach as WinkSwype. However, in the raycasting method, the cursor is replaced with a color-coded ray that changes from red (navigation) to green (gesture typing) during input. Additionally, since Gaze-Trigger relies on both eyes for input, it uses the standard calibration procedure rather than the single-eye calibration approach described earlier.

6.1 Participants

We initially recruited twenty participants for the user study. However, two were unable or uncomfortable winking one eye, so the final study was conducted with eighteen participants. Ten identified as female, and eight as male. Ages ranged from 22 to 40 years (M = 28.31 years, SD = 5.2). Two participants chose not to disclose their age. All participants reported native or bilingual proficiency in English. Each held at least a bachelor's degree. Four participants had experience with VR. Two were familiar with shape-writing on mobile devices, though neither used it as their primary input method. Sixteen participants were right-handed, two were left-handed. Most participants were right-eye dominant, while three demonstrating left-eye dominance. One participant wore corrective lenses and kept them on during the study. Each participant received US $15 as compensation for their time.

6.2 Apparatus

The study used an HTC Vive Pro Eye headset with one Base Station 2.0 sensor, which provides positional tracking within a 2 × 1.5 m area. The headset features dual 8.9 cm OLED displays with a resolution of 1440 × 1600 pixels per eye, offering a 110° FOV at a refresh rate of 90 Hz. It comes with two controllers, each equipped with a dual-stage trigger (Fig. 1b). The headset also includes a Tobii embedded eye tracker with 110° FOV and an accuracy ranging from 0.5° to 1.1° using the default five-point calibration. The tracker samples binocular gaze data at 120 Hz and provides real-time output, including timestamped gaze origin and direction data. The study was conducted using the experimental application (Fig. 1a), described in Section 3, which ran on a desktop computer with Windows 10 Enterprise, 32 GB RAM, an Intel Core i7 processor, a 1 TB hard drive, and an NVIDIA GeForce GTX 1080 GPU.



(a) (b)

Figure 2: (a) A participant performing the triangle test to determine eye dominance. (b) The calibration interface, where red spheres are shown sequentially as gaze targets. The screen is divided into nine zones to support spatial accuracy assessment during calibration.

6.3 Procedure

The study was conducted in a quiet research laboratory. Upon arrival, participants were greeted by a researcher who explained the study's goals and demonstrated the VR system and the three input methods. After the introduction, participants provided informed consent and completed a demographic questionnaire. Participants then performed a triangle test to determine eye dominance [13]. In this test, they extended their arms and formed a small triangle between their hands, focusing on a distant object through the opening (Fig. 2a). If the object remained centered upon closing the left eye, the right eye was considered dominant, and vice versa.

Following this, we calibrated the eye-tracking system. The process began with the built-in HTC Vive calibration tool using both eyes. This was followed by our custom single-eye calibration procedure (§4), during which participants were instructed to closed their non-dominant eye for improved precision of gaze tracking. During calibration and the rest of the study, participants sat comfortably in an adjustable chair positioned at an optimal distance from the tracking base station. After calibration, participants completed a training session with each input method, presented in counterbalanced order. For each method, they transcribed two pangram sentences (as described in Section 5.4), allowing them to become familiar with the techniques. The training lasted approximately six minutes. Once participants indicated that they were comfortable using the methods, testing began.

In the main session, participants transcribed the phrases drawn at random from a commonly used phrase set [41], divided into three blocks of five phrases each. One phrase was displayed at a time. Participants were instructed to transcribe each phrase as quickly and accurately as possible, and press Enter to proceed to the next phrase. While they were encouraged to correct any errors during input, doing so was optional. One-minute breaks were provided between blocks, and two-minute breaks were given between input methods. Participants were also offered up to three additional three-minute breaks upon request, although none made use of them.

At the conclusion of the input tasks, participants completed three questionnaires. First, they filled out the NASA-TLX, a workload assessment instrument that asks about their experience across six subscales: mental demand, physical demand, temporal demand, performance, effort, and frustration, each on a 20-point scale [34]. Next, they completed a usability questionnaire consisting of five 5-point Likert-scale items measuring perceived speed, accuracy, learnability, usability, and overall preference. Finally, participants responded to an eye fatigue questionnaire, adapted from prior work [19], using six 10-point Likert-scale items: difficulty seeing, strange feeling around the eyes, tired eyes, numbness, headache, and dizziness. The study then concluded with a debriefing session in which participants were invited to share open-ended feedback about their experience.

6.4 Design

The study used a 3 × 3 within-subjects design with the following independent variables and levels:

- Input Method (Gaze-Trigger, Raycasting, WinkSwype)
- Block (1, 2, 3)

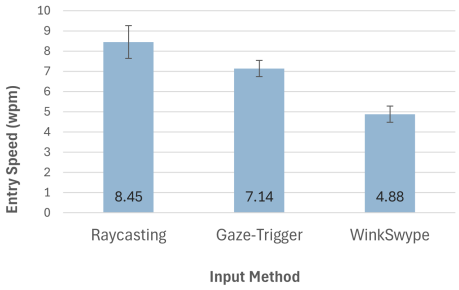
The order of the input methods was counterbalanced to mitigate learning effects. Each participant completed three blocks for each input method. A block consisted of five sentence transcription tasks selected from a corpus [41]. In total, 810 phrases were transcribed (18 participants × 3 methods × 3 blocks × 5 phrases/block). Using fewer phrases than keyboard-based techniques is common when evaluating gesture-based text input methods in VR (e.g., [33, 44]),

due to the greater physical effort, training, and demonstration required compared to traditional keyboard-based approaches. The full session lasted about 55 minutes per participant.

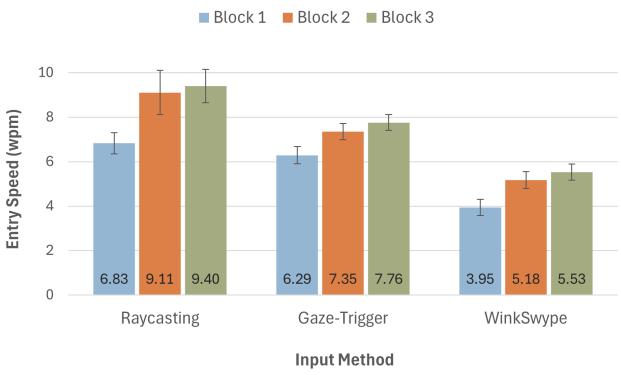
The dependent variables were text entry speed, error rate, and corrected error rate. Text entry speed was measured in words per minute (wpm) following the standard calculation [2]. Error rate was the percentage of incorrect words remaining in the final transcribed text per trial, while the corrected error rate reflected the percentage of errors made and subsequently corrected during a trial [38].

7 Results

A Shapiro-Wilk test confirmed normality of residuals, allowing for repeated-measures ANOVA with post hoc Tukey-Kramer tests for pairwise comparisons. We report eta-squared (η^2) for significant effects, with $\eta^2 \geq 0.14$ considered large [1]. For ordinal questionnaire data, we used the Friedman test, reporting Kendall's W for significant results, where $W \geq 0.5$ indicates a large effect [1].



(a) Entry Speed



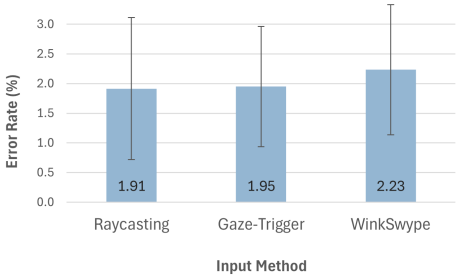
(b) Entry Speed × Block

Figure 3: Text entry speed (wpm) by input method (a) and across blocks (b). Error bars show ±1 standard error.

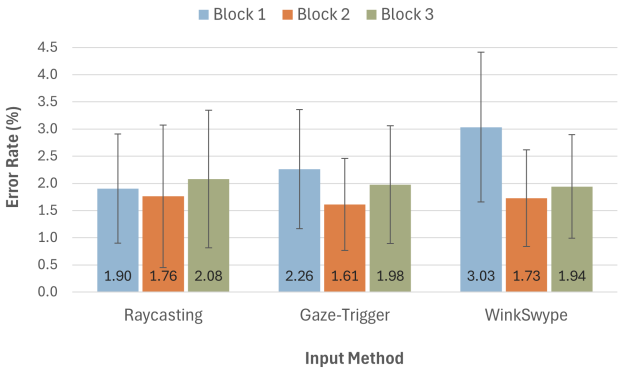
7.1 Text Entry Speed

Raycasting demonstrated the fastest text entry speed, averaging 8.45 wpm, followed by Gaze-Trigger at 7.14 wpm, and WinkSwype at 4.88 wpm (Fig. 3a). The effect of input method on text entry speed was statistically significant ($F_{2,17} = 98.28, p < .000, \eta^2 = 0.47$). A post hoc Tukey-Kramer test confirmed that all three input methods were significantly different from each other.

7.1.1 Block and Method Interaction. Entry speed across blocks for all input methods closely followed the power law of practice ($R^2 > 0.90$). An ANOVA revealed a significant interaction effect between input method and block on text entry speed ($F_{4,68} = 3.01, p < .05, \eta^2 = 0.01$). This interaction indicates that the improvements in text entry speed over time varied depending on the input method. Although all methods showed an increase in speed with practice, the rate of improvement was not uniform. Raycasting exhibited the most substantial gain, increasing from 6.83 wpm in Block 1 to 9.40 wpm in Block 3. Gaze-Trigger also showed steady improvement, rising from 6.29 wpm to 7.76 wpm. WinkSwype, while achieving the lowest overall speed, still demonstrated clear gains, from 3.95 wpm in Block 1 to 5.53 wpm in Block 3. Fig. 3b summarizes text entry speeds across blocks for all three input methods.



(a) Error Rate



(b) Error Rate × Block

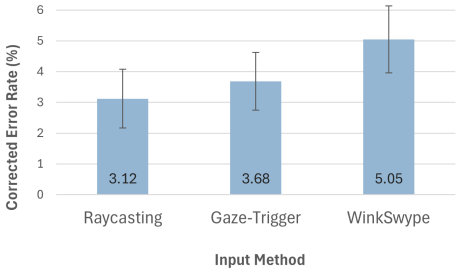
Figure 4: Error rate (%) by input method (a) and across blocks (b). Error bars show ±1 standard error.

7.2 Error Rate

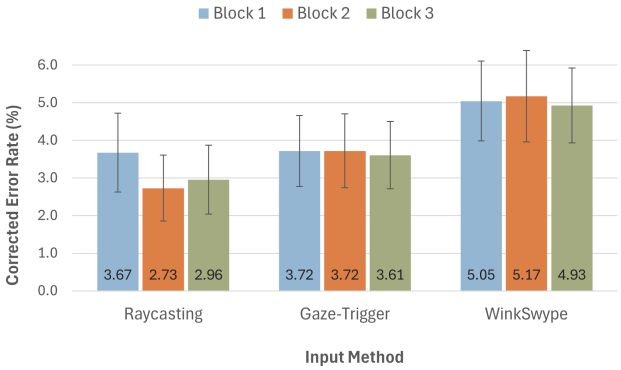
All methods yielded low error rates, each below 2% (Fig. 4a). Raycasting had the lowest error rate at 1.91%, followed by Gaze-Trigger (1.95%) and WinkSwype (2.23%). These differences were not statistically significant ($F_{2,17} = 0.75, p = .48$).

7.2.1 Block and Method Interaction. An ANOVA found no statistically significant interaction effect between the input method and the experimental block on error rate ($F_{4,68} = 0.63, p = .64$). Fig. 4b summarizes error rates across blocks. This suggests that the changes in error rates across the blocks followed a similar pattern for all

input methods. WinkSwype began with the highest error rate in Block 1 at 3.03%, which then decreased to 1.73% in Block 2 and settled at 1.94% in Block 3. Gaze-Trigger and Raycasting showed comparable trends, with error rates peaking in the first block and decreasing in subsequent ones.



(a) Corrected Error Rate



(b) Corrected Error Rate × Block

Figure 5: Corrected error rate by input method (a) and across blocks (b). Error bars show ±1 standard error.

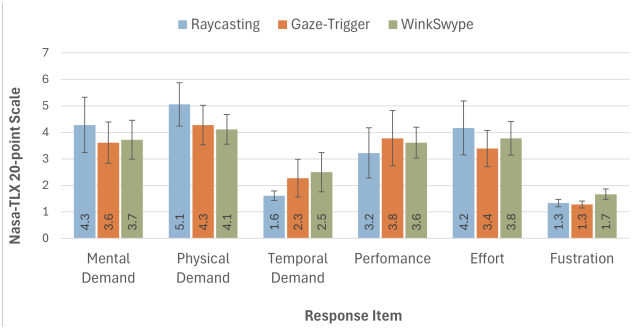
7.3 Corrected Error Rate

WinkSwype displayed the highest corrected error rate at 5.05%, followed by Gaze-Trigger at 3.68%, and Raycasting at 3.12% (Fig. 5a). The effect of input method on corrected error rate was statistically significant ($F_{2,17} = 28.17, p < .0001, \eta^2 = 0.17$). A post hoc Tukey-Kramer test revealed that WinkSwype's corrected error rate was significantly higher than those of the Raycasting and Gaze-Trigger input methods.

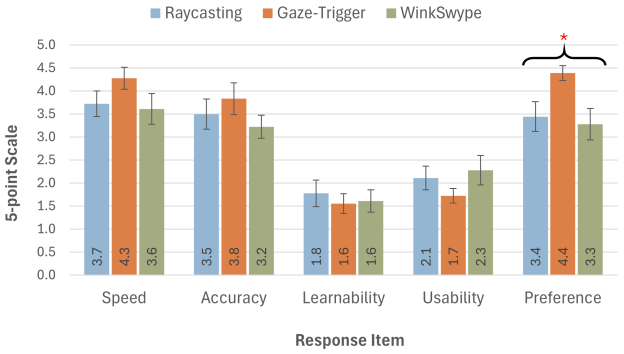
7.3.1 Block and Method Interaction. An ANOVA found no statistically significant interaction effect between the input method and the experimental block on corrected error rate ($F_{4,68} = 0.56, p = .69$). Fig. 5b shows a summary of the error rate by experimental block. This indicates that the rate at which users corrected their mistakes did not significantly differ across the blocks from one method to another. WinkSwype had the most corrections, with corrected error rates of 5.05% in Block 1, 5.17% in Block 2, and 4.93% in Block 3. Whilst, Raycasting and Gaze-Trigger had lower corrected error rates that also slightly decreased over time. Raycasting's rate went from 3.67% to 2.96%, and Gaze-Trigger's from 3.72% to 3.61% across the three blocks.

7.4 Perceived Workload

Gaze-Trigger yielded the best overall perceived workload ratings, followed by WinkSwype, with Raycasting performing the worst. On average, Raycasting was rated as more mentally and physically demanding, and required the most effort. Gaze-Trigger received the lowest performance ratings, while WinkSwype was perceived as the most temporally demanding and frustrating. However, these differences were not statistically significant. A Friedman test did not find a significant effect on mental demand ($\chi^2 = 0.17, p = .84$), physical demand ($\chi^2 = 0.50, p = .61$), performance ($\chi^2 = 0.10, p = .90$), effort ($\chi^2 = 0.24, p = .79$), temporal demand ($\chi^2 = 0.59, p = .56$), or frustration ($\chi^2 = 1.72, p = .19$). Fig. 6a presents the average perceived workload ratings for all methods.



(a) Perceived Workload



(b) Perceived Usability

Figure 6: (a) Average perceived workload ratings across input methods on a raw 20-point NASA-TLX scale (lower is better). (b) Average perceived usability ratings on a 5-point Likert scale (higher is better). Error bars show ±1 standard error.

7.5 Perceived Usability

The average usability ratings across all input methods are summarized in Fig. 6b. A Friedman test found no significant effect of input method on perceived speed ($\chi^2 = 1.55, p = .22$), perceived accuracy ($\chi^2 = 0.98, p = .38$), difficulty to learn ($\chi^2 = 0.21, p = .81$), or difficulty to use ($\chi^2 = 1.26, p = .29$). However, a significant effect

was found for user preference ($\chi^2 = 4.32, p < .05, W = 0.24$), with Gaze-Trigger being the most preferred method. These findings suggest that while most usability metrics were comparable across input methods, participants showed a clear preference for Gaze-Trigger.

7.6 Perceived Visual Fatigue

Although WinkSwype was associated with slightly higher ratings for difficulty seeing and tired eyes, a Friedman test revealed no statistically significant effect on any of the six visual fatigue measures: difficulty seeing ($\chi^2 = 0.70, p = .50$), strange feeling around eyes ($\chi^2 = 0.37, p = .69$), tired eyes ($\chi^2 = 2.66, p = .08$), numbness around eyes ($\chi^2 = 0.76, p = .47$), headache ($\chi^2 = 0.85, p = .43$), or dizziness ($\chi^2 = 0.48, p = .62$). Fig. 7 summarizes the average visual fatigue ratings for all methods.

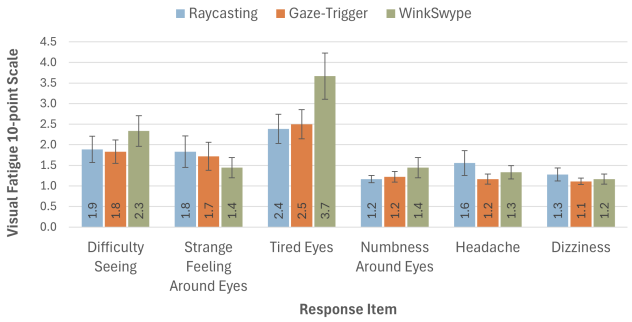


Figure 7: Average perceived visual fatigue ratings across input methods on a 10-point Likert scale (lower is better). Error bars represent ±1 standard error.

8 Discussion

Raycasting demonstrated the fastest text entry speed. Participants were generally comfortable using it and did not exhibit performance degradation due to the Heisenberg effect. Participants retained about 58% of their Raycasting text entry speed and 68% of their Gaze-Trigger speed when using WinkSwype. This is promising, given that WinkSwype is a fully hands-free method that requires no additional physical actions, making it potentially suitable for users with motor impairments. While WinkSwype was significantly slower than the other two methods, a result that is unsurprising given that the participants were non-disabled and could perform conventional controller-based inputs, all methods demonstrated clear improvements with practice. Notably, in the final block, participants retained 59% of Raycasting and 71% of Gaze-Trigger entry speed, indicating substantial gains from the first block and suggesting that WinkSwype performance will improve further with extended use.

WinkSwype also yielded slower entry speeds compared to some existing gaze-based techniques (§2). This we attribute to two key factors. First, most prior methods employ physical framing actions, such as button presses or dwell time, to signal gesture initiation and completion, while WinkSwype does not. Second, WinkSwype uses a significantly smaller keyboard, with approximately 70% smaller visual angle per key than those in previous studies. Larger keys typically afford greater gesture precision and lower error rates due to broader target areas. By contrast, WinkSwype's more compact keyboard is better suited to immersive VR applications, where large interfaces may obstruct the field of view, making our approach more practical despite its impact on speed.

All methods yielded low error rates (about 2% in the final block). WinkSwype's comparable error rate to Gaze-Trigger, despite using only one eye, provides indirect validation for the single-eye calibration model developed in this work. This suggests that participants could trace over the virtual keys with similar accuracy to traditional dual-eye calibration systems. Moreover, since all three methods employed the same hybrid gesture recognition model introduced in this study, the comparable accuracy across input types further validates its robustness for both gaze- and controller-based input. Interestingly, participants corrected significantly more errors using WinkSwype than with Raycasting (38% fewer corrections) and Gaze-Trigger (27% fewer). However, by the final block, this gap narrowed, and the corrected error rates for the two gaze-based methods were similar (4% and 5%, respectively). These findings suggest that while WinkSwype initially required more correction effort, affecting entry speed, this effect may diminish with practice.

Workload analysis revealed no significant differences in mental, physical, or temporal demand across input methods. In post-study debriefings, participants described WinkSwype as more engaging and less physically fatiguing than Raycasting, which involved prolonged arm use. However, these observations are based on a short study, and the workload effects during extended use remain unknown, warranting further investigation. Participants generally preferred the Gaze-Trigger method, appreciating its combination of gaze-based pointing with controller-based framing. Interestingly, no significant differences were observed in visual fatigue across the three methods. Participants rated WinkSwype slightly higher in fatigue, possibly due to prolonged use of a single eye for input. Although this difference was not statistically significant, it may be influenced by the small sample size and short duration of the study, thus further investigation is needed to assess potential effects during extended use.

8.1 Limitations

We acknowledge several limitations of this work. Although a core motivation for developing WinkSwype was to support users with motor impairments, our study did not include participants from this population. Despite extended efforts over the course of a year, we were unable to recruit individuals with motor impairments due to the limited pool of potential participants in our small-town location. Additionally, our participant sample lacked diversity in age, cultural background, and disability status, making it difficult to assess the generalizability of our findings to broader populations. While results from non-disabled participants showed that WinkSwype enabled users to retain over 50% of their entry speed compared to conventional methods, it remains uncertain whether these findings apply to users with motor impairments or to more diverse user groups. Future studies with representative populations are essential to validate WinkSwype's accessibility and effectiveness in real-world contexts.

The study used a relatively small number of phrases (15 per condition). While this choice helped keep the study duration manageable given the effort required for VR text entry, the limited sample size may affect the generalizability of the results. Similarly, pilot studies were conducted with small samples. Future work should incorporate larger and more diverse text corpora and recruit a larger participant pool to better evaluate performance across varied linguistic contexts.

Another limitation concerns the wink gesture itself. Two of the initial twenty participants (10%) were unable to naturally wink one eye and were excluded from the study. Although no large-scale data exist on this ability, informal estimates suggest that 60-70% of adults can perform a unilateral wink [27, 35]. While prior work and anecdotal evidence indicate that most individuals can learn to wink with practice [35], this requirement may limit the immediate usability of WinkSwype for some users and suggests a brief training phase may be necessary for broader adoption.

Finally, the current implementation and evaluation focused exclusively on English. Although the shape-writing principle could be extended to other languages, particularly those with more complex writing systems, doing so would likely require more sophisticated language modeling to manage ambiguity. Adapting WinkSwype for multilingual use remains an important direction for future work.

9 Conclusion & Future Work

We introduced WinkSwype, a novel hands-free, wink-based shape-writing technique for text entry in VR. Unlike existing methods that rely on external hardware or continuous physical actions, WinkSwype enables users to input text using only gaze and winking, framing gestures by closing and reopening one eye. To support this interaction model, we developed a single-eye calibration model and a hybrid gesture recognition algorithm optimized for gaze input. Results from a user study show that users retained 58-71% of their text entry speed compared to conventional controller-based methods, with comparable error rates and usability. Participants reported no significant increase in workload or visual fatigue during short-term use and found the method engaging and less physically demanding.

Future work will focus on evaluating WinkSwype with individuals with motor impairments to assess its real-world applicability and accessibility in assistive contexts. We also aim to investigate adaptive calibration techniques to improve the robustness and accuracy of single-eye gaze tracking across varying user conditions. Additionally, we plan to study the long-term learnability, usability, and potential fatigue associated with extended use of wink-based text entry. Finally, we intend to explore the optimal size and layout of virtual keyboards for gesture typing in VR, balancing gesture accuracy, visual comfort, and screen real estate.

References

[1] Ahmed Sabbir Arif. 2017. *A Brief Note on Selecting and Reporting the Right Statistical Test.* Technical Report. University of California, Merced, United States. https://www.theiilab.com/notes/HypothesisTesting.html

[2] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2009. Analysis of text entry performance metrics. In *Proceedings of the IEEE International Conference Science and Technology for Humanity – TIC-STH '09.* IEEE, New York, 100–105. https://doi.org/10.1109/TIC-STH.2009.5444533

[3] Doug Bowman, Chadwick Wingrave, Joshua Campbell, and Vinh Ly. 2002. *Using Pinch Gloves™ for Both Natural and Abstract Interaction Techniques in Virtual Environments.* Technical Report 547. Virginia Tech, Blacksburg, VA, USA.

[4] Doug A. Bowman. 2020. Embracing physical keyboard for virtual reality. *IEEE Annals of the History of Computing* 53, 09 (Sept. 2020), 9–10. https://doi.org/10.1109/MC.2020.3004605

[5] Doug A. Bowman, Christopher J. Rhoton, and Marcio S. Pinho. 2002. Text Input Techniques for Immersive Virtual Environments: An Empirical Comparison. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 46. SAGE, Thousand Oaks, CA, 2154–2158. https://doi.org/10.1177/154193120204602611

[6] 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 ACM SIGCHI Conference on Human Factors in Computing Systems – CHI EA '19.* ACM, New York, 1–6. https://doi.org/10.1145/3290607.3312762

[7] HTC Corporation. 2025. VIVE Pro Eye Specs & User Guide - Developer Resources. https://developer.vive.com/resources/hardware-guides/vive-pro-eye-specs-user-guide/

[8] Joshua Corvinus. 2016. VR keyboard: Demonstration of a touch-type virtual keyboard in VR. https://developer-archive.leapmotion.com/gallery/vr-keyboard

[9] Wenzhe Cui, Rui Liu, Zhi Li, Yifan Wang, Andrew Wang, Xia Zhao, Sina Rashidian, Furqan Baig, Iv Ramakrishnan, Fusheng Wang, and Xiaojun Bi. 2023. GlanceWriter: Writing Text by Glancing Over Letters with Gaze. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems.* ACM, Hamburg Germany, 1–13. https://doi.org/10.1145/3544548.3581269

[10] Tafadzwa Joseph Dube and Ahmed Sabbir Arif. 2019. Text entry in virtual reality: A comprehensive review of the literature. In *Proceedings of the 21st International Conference on Human-Computer Interaction – HCII '19 (LNCS 11567).* Springer, Cham, 419–437. https://doi.org/10.1007/978-3-030-22643-5_33

[11] Tafadzwa Joseph Dube, Kevin Johnson, and Ahmed Sabbir Arif. 2022. Shapeshifter: Gesture typing in virtual Reality with a force-based digital thimble. In *Extended Abstracts of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI EA '22.* ACM, New York, 230.1–230.9. https://doi.org/10.1145/3491101.3519679

[12] 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 '19.* IEEE, New York, 289–300. https://doi.org/10.1109/ISMAR.2019.00027

[13] Heiting Gary. 2019. Dominant Eye Test: How to Find Your Dominant Eye. https://www.allaboutvision.com/eye-care/eye-tests/dominant-eye-test/

[14] Jens Grubert, Eyal Ofek, Michel Pahud, and Per Ola Kristensson. 2020. *Back to the future: Revisiting mouse and keyboard interaction for HMD-based immersive analytics.* Technical Report. Cornell University. https://doi.org/10.48550/arXiv.2009.02927

[15] Jens Grubert, Lukas Witzani, Eyal Ofek, Michel Pahud, Matthias Kranz, and Per Ola Kristensson. 2018. Effects of hand representations for typing in virtual Reality. In *Proceedings of the IEEE Conference on Virtual Reality and 3D User Interfaces – 3DUI '18.* IEEE, New York, 151–158. https://doi.org/10.1109/VR.2018.8446250

[16] 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 – 3DUI '18.* IEEE, New York, 159–166. https://doi.org/10.1109/VR.2018.8446059

[17] Aleesha Hamid and Per Ola Kristensson. 2024. 40 years of eye typing: Challenges, gaps, and emergent strategies. *Proc. ACM Hum.-Comput. Interact.* 8, ETRA (May 2024), 222:1–222:19. https://doi.org/10.1145/3655596

[18] Ali Hassan, Gunho Sohn, and I. Scott MacKenzie. 2023. Comparison of one-handed and two-handed text entry in virtual reality using handheld controllers. In *Proceedings of the 14th International Conference on Applied Human Factors and Ergonomics – AHFE '23.* AHFE International, New York, 101–111. https://doi.org/10.54941/ahfe1003872

[19] H Heuer, G Hollendiek, H Kröger, and T Römer. 1989. Rest Position of the Eyes and Its Effect on Viewing Distance and Visual Fatigue in Computer Display Work. *Zeitschrift fur experimentelle und angewandte Psychologie* 36, 4 (Jan. 1989), 538–566.

[20] Josh Kaufman. 2012. 10,000 Most Common English Words. https://github.com/first20hours/google-10000-english

[21] Pascal Knierim, Thomas Kosch, Johannes Groschopp, and Albrecht Schmidt. 2020. Opportunities and challenges of text input in portable virtual reality. In *Extended Abstracts of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI EA '20.* ACM, New York, 1–8. https://doi.org/10.1145/3334480.3382920

[22] Pascal Knierim, Valentin Schwind, Anna Maria Feit, Florian Nieuwenhuizen, and Niels Henze. 2018. Physical keyboards in virtual reality: Analysis of typing performance and effect of avatar hands. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '18.* ACM, New York, 345.1–345.9. https://doi.org/10.1145/3173574.3173919

[23] Per-Ola Kristensson and Shumin Zhai. 2004. SHARK2: A large vocabulary shorthand writing system for pen-based bomputers. In *Proceedings of the 17th Annual*

AC Symposium on User Interface Software and Technology – UIST '04. ACM, New York, 43–52. https://doi.org/10.1145/1029632.1029640

[24] Chandan Kumar, Ramin Hedeshy, I. Scott MacKenzie, and Steffen Staab. 2020. TAGSwipe: Touch Assisted Gaze Swipe for text entry. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '20*. ACM, New York, 190.1–190.12. https://doi.org/10.1145/3313831.3376317

[25] Andrew Kurauchi, Wenxin Feng, Ajjen Joshi, Carlos Morimoto, and Margrit Betke. 2016. EyeSwipe: Dwell-free text entry using gaze paths. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '16*. ACM, New York, 1952–1956. https://doi.org/10.1145/2858036.2858335

[26] Jiaye Leng, Zijun Wang, Jian Wu, and Lili Wang. 2025. LipText: Lip Tracking Based Text Entry in VR. In *Extended Reality*, Weitao Song, Frank Guan, Shuai Li, and Guofeng Zhang (Eds.). Springer Nature, Singapore, 162–178. https://doi.org/10.1007/978-981-96-3679-2_11

[27] Chou-Ching K. Lin, Kuo-Jung Lee, Chih-Hsu Huang, and Yung-Nien Sun. 2019. Cerebral Control of Winking Before and After Learning: An Event-Related fMRI Study. *Brain and Behavior* 9, 12 (Nov. 2019), e01483. https://doi.org/10.1002/brb3.1483

[28] Yi Liu, Chi Zhang, Chonho Lee, Bu-Sung Lee, and Alex Chen. 2015. GazeTry: Swipe text typing using gaze. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction – OzCHI '15*. ACM, New York, 192–196. https://doi.org/10.1145/2838739.2838804

[29] Xueshi Lu, Difeng Yu, Hai-Ning Liang, Wenge Xu, Yuzheng Chen, Xiang Li, and Khalad Hasan. 2020. Exploration of hands-free text entry techniques for virtual reality. In *2020 IEEE International Symposium on Mixed and Augmented Reality – ISMAR '20*. IEEE, New York, 344–349. https://doi.org/10.1109/ISMAR50242.2020.00061

[30] Mathias N. Lystbæk, Ken Pfeuffer, Jens Emil Sloth Grønbæk, and Hans Gellersen. 2022. Exploring gaze for assisting freehand selection-based text entry in AR. *Proc. ACM Hum.-Comput. Interact.* 6, ETRA (May 2022), 141.1–141.16. https://doi.org/10.1145/3530882

[31] 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 *Proceedings of the 23rd International Conference on Intelligent User Interfaces – IUI '18*. ACM, New York, 263–267. https://doi.org/10.1145/3172944.3172988

[32] Xuanru Meng, Wenge Xu, and Hai-Ning Liang. 2022. An Exploration of Hands-free Text Selection for Virtual Reality Head-Mounted Displays. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, Singapore, 74–81. https://doi.org/10.1109/ISMAR55827.2022.00021 ISSN: 1554-7868.

[33] Aunnoy K Mutasim, Mohammad Raihanul Bashar, Christof Lutteroth, Anil Ufuk Batmaz, and Wolfgang Stuerzlinger. 2025. There Is More to Dwell Than Meets the Eye: Toward Better Gaze-Based Text Entry Systems With Multi-Threshold Dwell. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–18. https://doi.org/10.1145/3706598.3713781

[34] NASA. 1986. *NASA Task Load Index (TLX) V 1.0: Paper and Pencil Package*. Technical Report. Human Performance Research Group, NASA Ames Research Center, Moffett Field, CA, USA. https://humansystems.arc.nasa.gov/groups/TLX/downloads/TLX.pdf

[35] Lyelle L. Palmer. 1976. Inability to Wink an Eye and Eye Dominance. *Perceptual and Motor Skills* 42, 3 (June 1976), 825–826. https://doi.org/10.2466/pms.1976.42.3.825

[36] 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 – ETRA '18*. ACM, New York, 15.1–15.10. https://doi.org/10.1145/3204493.3204541

[37] 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 – APCHI '13*. ACM, New York, 215–220. https://doi.org/10.1145/2525194.2525288

[38] R. William Soukoreff and I. Scott MacKenzie. 2004. Recent developments in text-entry error rate measurement. In *Extended Abstracts of the ACM SIGCHI Conference Human Factors in Computing Systems – CHI EA '04*. ACM, New York, 1425–1428. https://doi.org/10.1145/985921.986081

[39] Marco Speicher, Anna Maria Feit, Pascal Ziegler, and Antonio Krüger. 2018. Selection-based text entry in virtual reality. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '18*. ACM, New York, 647.1–647.13. https://doi.org/10.1145/3173574.3174221

[40] Shota Vashakmadze. 2020. Modeling the line: Bresenham's algorithm, 1962–87. *Architectural Theory Review* 24, 3 (Sept. 2020), 262–278. https://doi.org/10.1080/13264826.2021.1930320

[41] Keith Vertanen and Per Ola Kristensson. 2011. A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*. ACM, New York, 295–298. https://doi.org/10.1145/2037373.2037418

[42] Gabriel Vivó-Truyols and Peter J. Schoenmakers. 2006. Automatic selection of optimal Savitzky-Golay smoothing. *Analytical Chemistry* 78, 13 (July 2006), 4598–4608. https://doi.org/10.1021/ac0600196

[43] Jacob O. Wobbrock, James Rubinstein, Michael W. Sawyer, and Andrew T. Duchowski. 2008. Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications (ETRA '08)*. ACM, New York, 11–18. https://doi.org/10.1145/1344471.1344475

[44] 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 (May 2019), 1991–2001. https://doi.org/10.1109/TVCG.2019.2898736

[45] 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 ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '17*. ACM, New York, 4479–4488. https://doi.org/10.1145/3025453.3025964

[46] Maozheng Zhao, Alec M. Pierce, Ran Tan, Ting Zhang, Tianyi Wang, Tanya R. Jonker, Hrvoje Benko, and Aakar Gupta. 2023. Gaze speedup: Eye gaze assisted gesture typing in virtual reality. In *Proceedings of the 28th International Conference on Intelligent User Interfaces – IUI '23*. ACM, New York, 595–606. https://doi.org/10.1145/3581641.3584072