SwEYEpinch: Exploring Intuitive, Efficient Text Entry for Extended Reality via Eye and Hand Tracking

Ziheng 'Leo' Li*
zihengleoli@cs.columbia.edu
Department of Computer Science, Columbia University
New York, NY, USA

Xichen He*
xh2623@columbia.edu
Department of Computer Science, Columbia University
New York, NY, USA

Mengyuan 'Millie' Wu
mw3209@columbia.edu
Department of Computer Science, Columbia University
New York, NY, USA

Zeyi Tong
zt2373@columbia.edu
Department of Computer Science, Columbia University
New York, NY, USA

Haowen Wei
hw2892@columbia.edu
Department of Computer Science, Columbia University
New York, NY, USA

Benjamin Yang
by2297@columbia.edu
Department of Computer Science, Columbia University
New York, NY, USA

Steven Feiner
feiner@cs.columbia.edu
Department of Computer Science, Columbia University
New York, NY, USA

Paul Sajda
ps629@columbia.edu
Department of Biomedical Engineering, Columbia University
New York, NY, USA

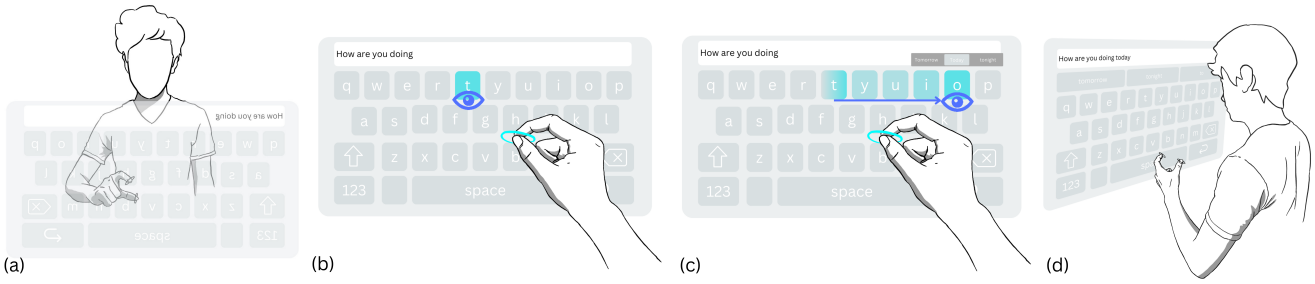


Figure 1: Typing in XR with *SwEYEpinch*. (a) The user (Head-Worn Display not shown) is about to type the word "today". (b) They look at the first letter "t" and start pinching. (c) Keeping fingers pinched, they swipe their gaze to "o". The system predicts likely word candidates based on the current gaze trace and preceding context, "How are you doing". (d) The user releases the pinch to confirm the first candidate "today", which is placed in the text box automatically. The other candidates are displayed above the keyboard and the user may choose one by looking at it and pinching.

ABSTRACT

Despite steady progress, text entry in Extended Reality (XR) often remains slower and more effortful than typing on a physical keyboard or touchscreen. We explore a simple idea: use gaze to swipe through a virtual keyboard for the fast, low-effort *where* and a manual pinch held throughout the swipe for the *when*, extending and validating it through a series of user studies. We first show that a basic version including a low-latency decoder with spatiotemporal Dynamic Time Warping and fixation filtering outperforms selecting individual keys sequentially, either by finger tapping each or gazing at each while pinching. We then add mid-swipe prediction and in-gesture cancellation, improving words per minute (WPM) without hurting accuracy. We show that this approach is faster and more preferred than previous gaze-swipe approaches, finger tapping with prediction, or hand swiping with the same additions. Furthermore, a seven-day 30-session study demonstrates sustained learning, with peak performance reaching 64.7 WPM.

*These authors contributed equally to this work.

CHI '26, April 13–17, 2026, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/2026/04
https://doi.org/10.1145/3772318.3791820

CCS CONCEPTS

• **Human-centered computing** → **Text input**; **Mixed / augmented reality**; **Gestural input**; *Pointing*.

KEYWORDS

Hand-tracked text entry; gaze-tracked text entry; extended-reality text entry; longitudinal user studies

ACM Reference Format:

Ziheng 'Leo' Li, Xichen He, Mengyuan 'Millie' Wu, Zeyi Tong, Haowen Wei, Benjamin Yang, Steven Feiner, and Paul Sajda. 2026. SwEYEpinch: Exploring Intuitive, Efficient Text Entry for Extended Reality via Eye and Hand Tracking. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 27 pages. https://doi.org/10.1145/3772318.3791820

1 INTRODUCTION

Efficient text entry in Extended Reality (XR) remains a significant challenge, impeding the widespread adoption of Head-Worn Displays (HWDs). While visual fidelity in XR has advanced rapidly, input methods have generally struggled to balance the trade-off between text entry rate, physical effort, and mobility.

Controller–raycast keyboards demand precise pointing at arm's length and yield modest typing speed (e.g., ∼15–17 WPM with ∼11% total error rate), with fatigue from sustained shoulder elevation and depth/occlusion ambiguity near key boundaries [2, 6]. Hand-based mid-air tapping avoids controllers but still lacks tactile feedback, amplifying "fat-finger" ambiguity on closely spaced keys [2]. External peripherals (physical keyboards) restore speed but break mobility and situational use, effectively tying users to a surface, or encumbering a hand if hand-worn [13, 21, 46]. Voice dictation can be fast, yet it fails in noisy or shared spaces and raises privacy concerns. Gaze, on the other hand, has clear appeal as a *targeting* signal in this landscape—eyes move quickly with very low muscular effort and eye tracking is increasingly available on HWDs [3, 9]—making it well-suited for text entry. Prior work on gaze-swipe typing has shown that users can naturally trace word-shaped paths with their eyes and that such "eye-gesture" input can outperform dwell-based gaze typing in entry speed [24]. However, gaze-only interaction typically relies on dwell-based selection for activation, which imposes an inherent speed cap and is susceptible to false activations—or on additional instrumentation (e.g., external touch surfaces or controllers [22, 56]), while hand-based activation is reliable but incurs a movement cost [2, 7, 30]. This motivates the desirability of a hybrid that preserves continuous, word-gesture *swipes* with gaze for rapid, low-effort *where*, while offloading the precise *when* to a manual pinch delimiter that marks word start and end. Our goal is an XR typing method that (1) approaches practical, everyday text entry rate, (2) reduces physical demand compared to mid-air tapping, and (3) avoids gaze-only pitfalls such as dwell latency and Midas touch [18].

We propose a hybrid interaction that bridges this gap: *SwEYEpinch*. We operationalize a simple principle: decouple targeting from commitment. The eyes are used exclusively to *trace* the word-gesture (fast, low effort), while a manual mid-air pinch is used to *delimit* the word (explicit, reliable). This preserves the speed of gaze and the reliability of manual input, without the fatigue of reaching for keys or the immobility of physical surfaces. To rigorously evaluate this approach against the spectrum of XR text entry, we present the design and evaluation of *SwEYEpinch* through four progressive user studies.

In an initial *User Study 1* (**US1**), we compare *SwEYEpinch-Basic*—a minimal gaze-swipe and pinch-delimiter variant that outputs word candidates upon pinch release—with two letter-by-letter baselines used in commercial XR headsets, focusing specifically on their interaction mechanics: *Finger-Tap* [19, 38, 51, 52] (Microsoft HoloLens, Meta Quest) and *Gaze&Pinch*, a well-studied selection technique in XR [20, 40, 44] (Apple Vision Pro).

Motivated by *US1*, we next ask whether *live, mid-swipe feedback* could further reduce effort and increase speed without compromising accuracy. The design must also mitigate eye fatigue from large in-view keyboards *and* reduce off-keyboard verification glances. To this end, we introduce *SwEYEpinch*, which continuously decodes words during the swipe, allowing users to see candidate words while tracing the letters rather than when they finish a swipe. We also add two safeguards to manage errors: *mid-swipe deletion* (allowing the user to cancel mid-swipe) and a *deletion peek window* (letting the user see what will be removed when they gaze at the delete button). These additions raise a natural question: *What impacts the user experience the most: live mid-swipe feedback, the pinch delimiter, or their combination?*

To answer this, *User Study 2* (**US2**) disentangles the effects of mid-swipe prediction and the delimiter (how the user signals the start/end of a word) by comparing *SwEYEpinch* against *SwEYEpinch-Basic* and two gaze-only swipe-based approaches with different delimiters: *SkiMR* [16], delimited by dwelling on the space key, and *GlanceWriter XR*, adapted for XR from its screen-based version [9], where the user starts and ends a word by moving their gaze in and out of the letter-key area. This isolates the contribution of *mid-swipe prediction* and of the *delimiter* itself. Our results show *SwEYEpinch* stands at the Pareto frontier in terms of typing speed and user preference while maintaining accuracy.

To test whether the gains observed in US2 hold up against strong baselines that users would encounter in commercial products, *User Study 3* (**US3**) evaluates *SwEYEpinch* against: *Finger-Tap with Prediction&Completion* and *Hand-Swipe* (mid-air word-gesture typing with hand ray and pinch delimiter [8, 10, 15, 36]; also available in Meta Quest) to the same level of word prediction and completion as *SwEYEpinch*.

Finally, *User Study 4* (**US4**) examines whether, with sustained use, users can gain proficiency fit for everyday use—how fast their performance improves, where they plateau, and what failure modes appear over time. We followed nine participants; each used *SwEYEpinch* for 30 sessions over seven consecutive days, allowing us to observe learning trends akin to what users would have had with daily exposure to the technique. By the end, all users achieved on average *above 40 WPM*, and three achieved *above 60 WPM*.

Our results support a simple design principle: *decouple targeting from commitment*. Use gaze only to *trace* the word (fast, low effort) and a manual pinch to *commit* (the delimiter); give the user immediate feedback with mid-swipe candidates, and reduce correction effort through mid-swipe deletion and a peek window. Together, these cut verification glances, remove dwell delays, and preserve explicit control.

We thus make the following contributions:

- We introduce *SwEYEpinch-Basic*, a gaze–gesture hybrid for XR text entry, and *SwEYEpinch*, an enhanced variant offering mid-swipe prediction, mid-swipe deletion, and a deletion peek window. These provide an optimized decoding pipeline (filtering by fixation detection combined with density-based clustering) that overcomes the high-frequency noise of eye-tracking data, enabling real-time mid-swipe feedback that was previously computationally prohibitive for gaze.
- Through three comparative studies, we benchmark *SwEYEpinch-Basic* and *SwEYEpinch* with other prominent XR text input methods. In **US1**, we establish that users prefer *SwEYEpinch-Basic* compared to traditional XR baselines, and gather feedback enabling us to design *SwEYEpinch*. In **US2**, we map the design space by isolating the roles of mid-swipe prediction and delimiter mechanics (*SwEYEpinch* vs. *SwEYEpinch-Basic* vs. gaze-only *SkiMR*/*GlanceWriter XR*). In **US3**, we compare *SwEYEpinch* against production-realistic contenders (*Finger-Tap w/ Prediction*, *Hand-Swipe*). For all user studies, we report on participants' speed, accuracy, and preference.
- Finally, we show that with repeated daily practice (**US4**), users of *SwEYEpinch* can achieve typing speeds comparable to a conventional keyboard (up to 64.7 WPM), highlighting its potential for everyday XR text-entry.

Code and Data Availability. Our decoding algorithm (Gaze2Word), and swipe trace data compiled from user studies are publicly available. See Section 3.2.1 for details.

2 RELATED WORK

We review prior work, from well-studied conventional methods used in commercial products to more recent developments. For a broader overview of XR text entry as a whole, see the comprehensive survey by Bhatia et al. [5].

2.1 Text-entry in HWDs—Conventional Methods and Alternatives

Controller and Hand-tracking. Many XR text-entry approaches rely on handheld controllers with virtual keyboards (point-and-click via raycasting) or mid-air typing with hand tracking [54]. For instance, one study reported 16.65 WPM with 11.05 total error rate (TER) using controller-based raycasting [6]. A more recent study [2] found 15.3 WPM with controllers, slightly outperforming hand tracking (13.8 WPM). Using 3D protruding virtual keys (keys with depth) can also improve typing speed (15.6 WPM with 3D keys vs 13.4 WPM with flat keys) and user experience [2]. Overall, controllers provide reliable selection and tactile feedback (via trigger clicks) but still yield low text entry rate compared to desktop typing; users retain 60% of their desktop performance [13], or equal it when typing with a conventional keyboard in XR [21].

Holding controllers can be cumbersome for on-the-go scenarios. Xu et al. note that while one can pair a physical keyboard with an AR display, it effectively locks the user to a stationary setup, whereas virtual keyboards (operated via hand gestures) better support on-the-go usage [55]. Hand-tracking and mid-air keyboards improve immersion by allowing direct interaction. Luong et al. report that while controllers were faster for distant pointing tasks, participants actually favored freehand interaction for direct mid-air selection tasks [30]. This implies that for text input styles where users "touch" keys in the air (a near-field task), hand tracking can be more natural and even preferred, but it can suffer from a lack of tactile feedback, leading to increased arm fatigue as reported in a few studies [2, 7].

Alternative Approaches. Alternative approaches integrate other bodily signals, encompassing a diverse range of interaction techniques that eliminate the need for physical controllers or hands, improving accessibility and reducing physical strain. Voice dictation in XR can be extremely fast in WPM and is already used in practice (e.g., voice-to-text on HoloLens). In a controlled study [1], speaking sentences aloud and then making corrections with hand-tracking averaged about 28 WPM (up to 36 WPM with perfect recognition) with low error rates (0.5%), far surpassing the 11 WPM of purely hand-based mid-air typing. However, voice is impractical in noisy or shared spaces.

Some hands-free typing experiments have tried using facial muscle movements to select letters, but suffer from very slow speeds. One comparison found that using head-gaze plus a "mouth-open" gesture to confirm each character yielded only about 3.07 WPM, and using an "eyebrow raise" yielded 2.85 WPM [12]. Other techniques integrate other body movements, such as NeckType, which allows users to select characters via subtle head gestures [29]. On the other hand, *Hummer* [14] uses vocal humming as a delimiter, offering a hands-free alternative, yet vocal input faces social acceptability challenges in public or quiet environments. Despite the variety of alternative hands-free input methods, gaze emerges as the most technically suitable modality for XR text entry. Eye movements are extremely fast and require minimal effort, involving very small, well-conditioned muscles, enabling rapid targeting of virtual keys or UI elements.

2.2 Gaze-based text entry

DwellType. Gaze-based text entry has been widely studied in XR as a hands-free alternative to traditional input. Among these methods, DwellType—where users select letters by fixating for a set duration—is one of the most common baselines due to its simplicity and accessibility [29]. However, its reliance on delayed activation fundamentally limits speed. Mackenzie et al. estimated a theoretical maximum of around 22 WPM (with a 0.5 s dwell and rapid 40 ms saccades to move between keys), yet novice users achieve only 8–10 WPM, and experienced users plateau around 20 WPM [33]. Comparative studies show DwellType (10.59 WPM) is significantly slower than TapType (15.58 WPM) and GestureType (19.04 WPM) [56], and offers the lowest throughput (1.12 bits/s) compared to pinch gestures(1.60 bits/s) or button click (1.88 bits/s) [40]. Although Yu et al. [56] evaluates head-gaze–based TapType and GestureType rather than eye-gaze methods, these techniques still outperform DwellType, reinforcing that the dwell-timer mechanism, rather than gaze modality, is the main limiting factor [39]. Furthermore, designers face a speed–accuracy trade-off: longer dwell durations are slow but reliable, while shorter ones risk many errors and unintentional activations [12]. Beyond performance, DwellType imposes high cognitive load and visual strain, with studies reporting increased blink frequency, altered saccade patterns, and session limits ($\leq$ 15 minutes) to mitigate fatigue [4, 26].

Eye-swipe Typing. To overcome dwell-time limitations, researchers have proposed dwell-free gaze typing. Kurauchi et al. [24] introduced *EyeSwipe*, adapting smartphone word-gesture keyboards [57] with reverse crossing for selection, achieving 11.7 WPM compared to 9.5 WPM with dwell-based typing. Similarly, Cui et al. [9] developed *GlanceWriter*, leveraging a probabilistic decoding algorithm to dynamically predict user intent, achieving a 67% speed improvement over EyeSwipe. In addition, Liu et al. combined gaze-tracking with the Moving Window String Matching algorithm for more accurate word entries [27]. These approaches demonstrate that eliminating dwell-time delays can significantly enhance gaze-based text entry. The adaptation of the swipe in an HWD by Yu et al. [56] uses head movement to activate the swipe instead of gaze, reaching 24.7 WPM after one hour of practice, while Lu et al. found that, in head-gesture typing, eye blinks were the most effective selection mechanism compared to dwell or swipe [28]. Other XR adaptations have explored using the *space key* as a delimiter to mitigate Midas-touch ambiguity [16] and relax spatial matching constraints for more ergonomic gaze-based typing [35]. Another study focusing on head vs. gaze pointing showed that users generally prefer gaze over head-based input, describing it as faster, more pleasant, and efficient [12]. A rich body of prior work has therefore investigated ways to improve the accuracy of gaze input more generally [23], including probabilistic correction and snap-to-target techniques.

Pure gaze-based techniques leave ample room for improvement in both WPM and accuracy due to their reliance on precise gaze trace and potential high visual strain [11, 24]. Studies indicate that hybrid approaches, combining gaze with additional input modalities such as hand gestures, head movements, or predictive models, can increase speed and usability [31]. While multimodal gaze-typing has been explored on desktop and mobile surfaces, its application in mobile XR presents unique constraints. *TagSwipe* [22] demonstrated that anchoring gaze swipes with a touch-surface delimiter significantly improves performance. However, TagSwipe relies on physical contact (touchscreen or trackpad), which limits user mobility in everyday XR scenarios.

2.3 Combining Hand Tracking and Gaze

Combining gaze with manual input has long been proposed as a natural division of labor: the eyes rapidly target, while the hand provides confirmation or fine-grained control. Early work such as Gaze-touch [42] demonstrated this principle on touchscreens, showing that gaze can efficiently pre-select targets for subsequent touch interaction. More recent XR studies extend this idea by using the hand as a pointing device while leveraging gaze implicitly to accelerate cursor movement [58], or by combining gaze selection with controller confirmation [45]. Extending this principle, Wagner et al. [53] analyzed eye–hand coordination for object manipulation in near-space XR, finding systematic gaze-leading-hand dynamics that can be harnessed for input design. These insights motivate text entry methods that combine gaze and hand actions, as in our SwEYEpinch approach.

Another promising direction is to integrate gaze with body-based gestures. HGaze exemplifies this by combining subtle head gestures with gaze-based swipe typing [11]. Pfeuffer et al. [43] outlined core design principles and challenges for gaze–pinch interaction in XR, emphasizing simplicity, immediacy, and clear feedback as critical factors for usable systems. Empirical findings echo these principles: Park et al. [41] showed that increasing gesture complexity (e.g., multi-step or compound gestures) significantly degrades performance compared to simpler gaze–pinch combinations. This suggests that, while gaze–pinch can be powerful, designers should keep gestures lightweight to maximize usability.

Freehand selection refined by gaze tracking has also been explored in XR. Lystbæk et al. [31] proposed S-Gaze&Finger and S-Gaze&Hand, where gaze assists freehand input to enhance accuracy and reduce hand movement. Their study found that S-Gaze&Finger reduced hand movement by over 50% compared to Finger-Tap (as in HoloLens 2), though the study showed no improvement in speed (10.7 WPM for S-Gaze&Finger and 11.4 for Finger-Tap). This reveals an interesting observation: the baseline techniques in commercial headsets if included in a study, have repeatedly been found hard to beat.

While gaze-only methods such as *SkiMR* [16] have been compared against other gaze-only techniques in XR, prior work has not examined gaze-based swipe techniques alongside commercial XR baselines such as Finger-Tap and Hand-Swipe, or hybrid approaches such as Gaze&Pinch. This leaves an open question of how gaze-based typing compares to the input techniques users already encounter in practice. Our work addresses this question by conducting a comparative analysis of these input techniques to identify the most effective methods for text entry in XR.

Moreover, most recent XR studies on gaze-based swipe input have focused on short-term use and have not assessed performance at the expert level. Earlier work shows that text-entry performance typically plateaus after extended training [37], with expert typists on physical keyboards eventually matching their real-world typing speeds in XR [21]. To address this critical gap, we conducted an extended longitudinal user study (US4), wherein nine participants performed 30 sessions (around 30 min per session, including setup) with the best-performing technique identified from US1, US2, and US3. This extensive practice allows us to examine expert-level typing performance, providing additional insight into the potential of gaze-based swipe techniques for text input.

Scope and Baseline Selection

Our experimental scope is constrained by two design goals: (1) use only sensors that are already integrated into many modern HWDs (eye tracking and hand tracking), and (2) study *silent*, everyday text entry that does not rely on external surfaces or voice. This excludes influential techniques such as TagSwipe [22] and GestureType [56], which require an external device for input, and Hummer [14], which uses humming as the primary activation channel. We instead treat these as *external reference points*, comparing reported WPM and learning trends, while reserving in-study baselines for techniques that can be implemented with HWD-only sensing under our apparatus.

3 DESIGN OF SWEYEPINCH: TINY-EXPLICIT GESTURE-DELIMITED GAZE SWIPE TYPING

Unlike existing XR text-entry methods that rely on dwell-based gaze selection (slow and error-prone) or swipe-only gaze gestures (ambiguous activation) [9, 11, 16, 22, 24, 35], *SwEYEpinch* offers two key innovations: a gaze–explicit gesture-combined input method and an optimized word-completion algorithm tailored for XR text entry. Prior work shows that mid-air gesture typing in XR imposes substantial visuomotor demands, requiring users to allocate their visual attention to the keyboard area and maintain tight eye–hand synchronization to compensate for the lack of physical boundaries [17]. *SwEYEpinch* mitigates these burdens by leveraging a simple pinch delimiter to explicitly confirm word input, which minimizes unintentional selections and reduces both cognitive and visual fatigue compared to prolonged dwell or gaze-only activation. Moreover, its backend prediction algorithm is optimized to decode words on the fly during a swipe. This combination of gesture-based control and predictive completion enables users to input text more fluidly, with reduced physical strain and fewer errors.

3.1 How SwEYEpinch Works

To understand the interaction loop of *SwEYEpinch*, we can begin by considering *Gaze&Pinch*, a popular XR text-entry technique that uses eye tracking to select individual keys, each confirmed with a pinch gesture. *SwEYEpinch* also leverages eye tracking and gestures. In smartphones, "swipe typing" [57] extends the traditional "tap key to type," and *SwEYEpinch*, together with other gaze-based swipe techniques, follows the same logic for gaze input: Instead of tapping each key individually, you can swipe across multiple letters while pinching your fingers. As noted by Yu et al. [56], swipe-enabled keyboards are advantageous because they let users quickly tap out a single letter or swipe an entire word, thus addressing the out-of-vocabulary problem. We refer to this baseline version as *SwEYEpinch-Basic*, to distinguish it from the *SwEYEpinch* variant introduced later in Section 6.1.

The *SwEYEpinch-Basic* interaction loop is straightforward. As illustrated in Figure 1, to type a word, the user first looks at the word's initial letter, then pinches their fingers and swipes their gaze through all intended letters. After traversing the letters, they release the pinch to end the swipe. A decoding algorithm (detailed in the next section) then suggests the top four most likely words. Note that Figure 1 shows the *SwEYEpinch* version (Section 6.1), where predictions appear on the fly, rather than only after releasing the pinch.

Once the user finishes swiping, the system automatically inserts the first candidate into the text box and shows the remaining three candidates right below it. Selecting any of these alternate candidates is done by gazing at it and pinching, just as if tapping a single key. If the user presses the delete key immediately following a swipe, it removes the entire last word rather than a single space. Once another input key (e.g., a letter, number, or space) is pressed, the delete key reverts to its normal behavior and removes only one character per press.

Seamless Mode Switching. Crucially, *SwEYEpinch-Basic* inherently supports character-level entry too [57]. If the user performs a *pinch-and-release* on a key without generating a swipe trajectory (i.e., without looking at other keys), the system registers a standard single-letter tap. This allows users to fluidly mix word-level swiping with character-level tapping (e.g., for out-of-vocabulary words or passwords) using the same modality, effectively combining the benefits of *SwEYEpinch* and *Gaze&Pinch* in a single interface.

3.2 SwEYEpinch Decoding Algorithm and Optimization

The *SwEYEpinch* decoding algorithm, which we call `Gaze2Word`, builds on previous work [9, 24, 56] and is described in 1. In essence, *Gaze2Word* takes a user's gaze trace as input and outputs the top-k candidate words by matching this gaze trace to template traces stored in a dictionary. The resulting similarity scores are then combined with an n-gram language model to generate a final ranked list of candidate words.

The Challenge of Real-Time Gaze Decoding. While swipe prediction is standard on mobile touchscreens, porting this to gaze presents a specific computational challenge: signal density. Touch events are discrete and relatively sparse. In contrast, eye tracking generates high-frequency data (200 Hz) laden with jitter and micro-saccades. A raw two-second gaze swipe contains ~400 points; running Dynamic Time Warping (DTW) on this raw stream for every frame causes latency that breaks the feedback loop.

To enable the *mid-swipe predictions* introduced in this work, we implement a specific optimization pipeline before the DTW step. We chain *Velocity-Threshold Identification* (I-VT) to remove saccades with *Density-Based Spatial Clustering* (DBSCAN) to reduce the gaze trace cardinality by over 90% (from ~320 to ~15 points per word, see Appendix A).

Below, we describe the core components of Algorithm 1 that utilize this pruned data, emphasizing the key optimizations that make on-the-fly decoding feasible (Section 6.1)—that is, displaying candidate words while the user is still swiping, rather than only at the end. In addition, we added a temporal axis to the DTW distance measure, improving decoding accuracy at the cost of a slight increase in latency. A post-hoc analysis using US1 data shows that, with the proposed optimization and spatial-temporal DTW, the proposed algorithm is more accurate and has lower latency when compared with decoders in prior work (Table 2). For more algorithm details, see the supplementary material.

Spatiotemporal Dynamic Time Warping. After we reduce the swipe trace, the next step is to match it with word template traces. We align the preprocessed gaze trace $\mathbf{g} = \{(x_i, y_i, t_i)\}_{i=1}^{n}$ with template word paths $\mathbf{q} = \{(x'_j, y'_j, t'_j)\}_{j=1}^{m}$ from $\mathcal{V}_{\text{filtered}}$, where q consists of the keyboard centers of the letters that make up this word. We represent each gaze point as (x_i, y_i, t_i), where (x_i, y_i) is the 2D position where the gaze hit the keyboard and t_i is its timestamp. Template points are defined by letter-center positions, with $t'_j = j$ serving as a positional index to enforce a weak forward-time prior while remaining robust to speed variation. Then we have the pointwise distance as

$$dist(\mathbf{g}_i, \mathbf{q}_j) = \sqrt{(x_i - x'_j)^2 + (y_i - y'_j)^2 + (t_i - t'_j)^2}.$$

Combining with Language Model. Lastly, we multiply $p_{\text{dist}}(w; \mathbf{g})$ with an n-gram language-model probability $p_{\text{ngram}}(w \mid \text{context})$.

Algorithm 1 Gaze2Word: spatiotemporal Dynamic Time Warping (DTW) on pruned swipe trace with n-gram fusion

Require: Raw gaze samples $\mathbf{g}_{raw} = \{(x_i, y_i, t_i)\}_{i=1}^{N}$, language context C, lexicon $\mathcal{V}$, keyboard geometry $\mathcal{K}$, top-k, weights $\alpha \in [0,1]$, $\epsilon > 0$, I-VT params θ_{ivt}, DBSCAN params $(\varepsilon, \text{minPts})$
Ensure: Ranked list TopK

$\mathbf{g} \leftarrow$ I-VT$(\mathbf{g}_{raw}; \theta_{\text{ivt}})$ ▷ Fixation detection; returns fixation points with timestamps
$\mathbf{g} \leftarrow$ DBSCAN_Reduce$(\mathbf{g}; \varepsilon, \text{minPts})$ ▷ Cluster and keep time-ordered centroids
$\mathcal{V}_{\text{filtered}} \leftarrow$ FilterCandidates$(\mathbf{g}[1], \mathcal{V}, \mathcal{K})$ ▷ Gate by first-fixation key proximity / length
if $\mathcal{V}_{\text{filtered}} = \emptyset$ **then**
    $\mathcal{V}_{\text{filtered}} \leftarrow$ LM prefix-trie shortlist from C
end if
for all $w \in \mathcal{V}_{\text{filtered}}$ **do**
    $\mathbf{q}_w \leftarrow$ TemplateTrace$(w, \mathcal{K})$ ▷ Key-center path; set $t'_j \leftarrow j$
    $\delta_w \leftarrow \text{DTW}_{xyz}(\mathbf{g}, \mathbf{q}_w)$ ▷ DP on distance $d((x,y,t),(x',y',t'))$
end for
$d_{\min} \leftarrow \min_w \delta_w, \quad d_{\max} \leftarrow \max_w \delta_w$
for all $w \in \mathcal{V}_{\text{filtered}}$ **do**
    $p_{\text{dist}}(w) \leftarrow 1 - \dfrac{\delta_w - d_{\min}}{(d_{\max} - d_{\min}) + 1 \times 10^{-12}}$ ▷ Distance→[0,1]
    $p_{\text{ng}}(w) \leftarrow p_{\text{ngram}}(w \mid C)$
    $s(w) \leftarrow \big(p_{\text{ng}}(w) + \epsilon\big)^{\alpha} \cdot \big(p_{\text{dist}}(w)\big)^{1-\alpha}$ ▷ Fusion (Sec. 3.2)
end for
return top-k words by $s(w)$ (break ties by higher p_{dist})

Let $\alpha \in [0,1]$ be a weighting parameter and ϵ a small smoothing term (we used $\epsilon = 1e-8$). Then,

$$P(w \mid \mathbf{g}, \text{context}) = \Big(p_{\text{ngram}}(w \mid \text{context}) + \epsilon\Big)^{\alpha} \times \Big(p_{\text{dist}}(w; \mathbf{g})\Big)^{1-\alpha}.$$

Here, α controls how much the n-gram model factors into the combined probability. Through pilot testing, we found $\alpha = 0.05$ to be most effective. We then rank the words in descending order of $P(w \mid \mathbf{g}, \text{context})$, retaining the top k as the final output.

3.2.1 Open-Source Implementation and Multi-Study Swipe Dataset. We provide an open-source Python implementation of the SwEYEpinch (`Gaze2Word`) algorithm as a PyPI package[1], to help researchers and developers build on our work. Computational bottlenecks such as DTW are wrapped in optimized, compiled extensions to meet real-time latency, enabling *SwEYEpinch* to refresh candidates whenever the user saccades to a new key. We additionally release a Unity demo package illustrating `Gaze2Word` integration in Unity applications, using the cursor as a proxy for gaze[2].

Dataset. We release a *headset-captured, word-level* gaze-swipe dataset[3] aggregated from all our user studies covering all swipe conditions (*SwEYEpinch-Basic*, *SwEYEpinch*, *Hand-Swipe*, *SkiMR*, and *GlanceWriter XR*) with recordings from 71 participants and 9248 word attempts. Each sample corresponds to one intended word (from delimiter onset to confirmation) and includes raw 200 Hz gaze, swipe hit points on the keyboard used for decoding, and the intended word. To our knowledge, this is the first public XR *word-level* gaze-swipe dataset of this scope.

[1]https://pypi.org/project/gaze2word/, https://github.com/SwEYEpinch/Gaze2Word
[2]https://github.com/SwEYEpinch/SwEYEpinch-Unity-Demo
[3]https://huggingface.co/datasets/SwEYEpinch/SwEYEpinch

4 USER STUDY METRICS AND APPARATUS

To compare text-entry methods, we report words per minute (WPM) and total error rate (TER). Following common practice, WPM normalizes by five-character words:

$$\text{WPM} = \frac{|S|/5}{t}, \tag{1}$$

where $|S|$ is the transcribed-string length (characters) and t is entry time (minutes). We compute TER following Soukoreff and MacKenzie [49]:

$$\text{TER} = \frac{IF + INF}{C + IF + INF}, \tag{2}$$

where C is correct keystrokes, IF incorrect-but-fixed, and INF incorrect-and-not-fixed. For learning across sessions, we report Learning Rate as the regression slope β in $\text{WPM}_s = \alpha + \beta s + \epsilon$, where s is session index.

Statistical analysis and counterbalancing. To mitigate longitudinal learning and fatigue effects, we employed a Staggered Latin Square to rotate condition orders systematically across sessions. Given the variation in participant counts across studies (unbalanced block sizes), we utilized **Linear Mixed Models (LMMs)** for all performance analyses rather than repeated-measures ANOVAs. We modeled *Technique* and *Presentation Order* as fixed effects, with *Participant* as a random intercept. Although perfect counterbalancing was not achievable due to sample-size constraints (e.g., uneven multiples of orderings), this approach allows us to statistically isolate intrinsic technique performance from the significant learning effects such as the ordering bias observed over the multi-day study. *Multiple comparisons.* All pairwise comparisons derived from the LMMs were adjusted using the Benjamini–Hochberg false discovery rate (FDR) method to control for expected false discoveries while maintaining power. All p-values reported in the results are adjusted values.

All studies used a Varjo XR-3 (200 Hz eye tracking) with integrated Ultraleap hand tracking in Unity 2022.3.10f1 on Windows 11 with an Intel® Core™ i9-10900K CPU @ 3.70GHz and an Nvidia GeForce RTX 3090 graphics card. PhysioLabXR 1.1 [25] is used to collect data and serve the decoder algorithm. The same QWERTY layout was used across studies (see the supplementary material). In the experiment scene, the keys were rectangular (width 51.0 mm, height 56.7 mm) with a horizontal gap of 11 mm and a vertical gap of 9.30 mm. The keyboard was positioned 70 cm from the user, corresponding to visual angles of roughly 4.18° horizontally and 4.64° vertically for each key and the entire keyboard subtends 47.9° in width and 35.6° in height, consistent with established parameters for gaze-based text-entry research. During practice, the keyboard distance was adjusted by up to ±15 cm for comfort. The 200 Hz eye tracker provided sufficient temporal resolution to capture rapid eye movements between keys without dropouts. Throughout all sessions, the application sustained rendering rates above 60 fps, ensuring stable visual performance and eliminating latency that could influence typing behavior. This apparatus and keyboard layout are identical across all user studies (US1–US4).

5 USER STUDY 1: FROM SIMPLE BASELINES TO SWEYEPINCH-BASIC

To test the design intuition *decouple targeting from commitment*, we began with a minimal hybrid, *SwEYEpinch-Basic* (§3.1) that decodes at the word-level: gaze swipe for fast targeting, and a pinch gesture to confirm the predicted word. In contrast, current commercial XR systems (e.g., Meta Quest *Finger-Tap*, Apple Vision Pro *Gaze&Pinch*) expose mainly non-predictive, character-level keyboards. We therefore use these widely deployed character-level techniques—without adding word prediction—as status-quo baselines for US1:

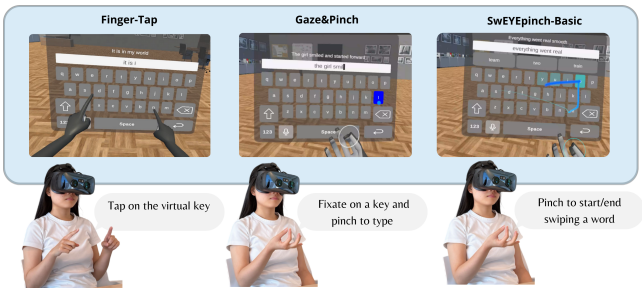


Figure 2: Techniques evaluated in US1: two simple XR baselines–*Finger-Tap* (letter-by-letter mid-air tapping on a virtual keyboard) and *Gaze&Pinch* (gaze-targeted key selection with a pinch delimiter)–compared against our proposed *SwEYEpinch-Basic*, our pinch-delimited gaze-swipe technique.

Finger-Tap. Users aim their tracked finger at the desired key and tap (Figure 2 left). This method relies on direct, physical mapping but requires significant arm movement.

Gaze&Pinch. This method proceeds similarly to tapping, except the user fixates their gaze on a target key and performs a "pinch and release" gesture (Figure 2 middle).

However, both the tracked fingertip and the gaze can easily land near key boundaries, introducing uncertainty over the user's intended letter. Because letter-by-letter typing is subject to this "fat-finger" problem, for both baselines, we compute the most probable letter by combining a 2D Gaussian centered at the tap/gaze location and a character-level n-gram (see Appendix B for details). In addition, recognizing the late-trigger problem often found with gaze-based inputs, we include an analysis of this issue in the supplementary material.

While this compares our predictive swipe technique against character-level tapping, this experimental design aligns with established prior work [22, 24, 56], which benchmarks novel predictive methods against standard non-predictive defaults to determine the practical performance delta over the status quo. Prior gaze-only or hand-only methods either incur dwell latency/Midas touch or high physical demand [2, 7, 30]. Although SwEYEpinch-Basic employs word prediction as part of its design, *Finger-Tap* and *Gaze&Pinch* do not natively include predictive assistance. In US1, we intentionally evaluate methods without adding additional prediction to the baselines, so that we could isolate the effect of interaction mechanics alone without confounding these results with differing levels of predictive support. Thus, US1 compares *SwEYEpinch-Basic* to these production-style XR baselines to test the following hypotheses:

- **H1.1.** After limited practice (Session 5), *SwEYEpinch-Basic* achieves higher text-entry speed (WPM) than *Finger-Tap* and *Gaze&Pinch*.
- **H1.2.** By Session 5, *SwEYEpinch-Basic* maintains non-inferior TER relative to *Finger-Tap* and *Gaze&Pinch*.
- **H1.3.** *SwEYEpinch-Basic* lies on or above the speed–preference Pareto frontier relative to both baselines; at similar speeds it is preferred more, and at similar preference it is faster.

5.1 Study Design

We used a within-subjects, five-session design (max one session/day). Each session began with two warm-up phrases per technique, then a blocked transcription task with three techniques: *Finger-Tap*, *Gaze&Pinch*, and *SwEYEpinch-Basic*. Participants transcribed 12 unique MacKenzie–Soukoreff phrases [32] per technique (36 per session), with technique order counterbalanced across participants and sessions. Phrases were not repeated for a given participant across sessions. After each session, participants completed a raw NASA Task Load Index (TLX) and provided a technique preference ranking with a brief justification.

5.2 Participants

Forty participants (ages 14 to 34, $\bar{x} = 24.5$; 23 male, 16 female, 1 non-binary) completed US1. All reported normal or corrected-to-normal vision and proficient English. Previous exposure to XR was common (62.5%), but only 10% had used XR keyboards before.

5.3 Results and Discussion

SwEYEpinch-Basic is faster than both baselines after five sessions, supporting H1.1. By Session 5, *SwEYEpinch-Basic* was faster than both baselines (Fig. 3a): 22.50 WPM vs. *Finger-Tap* 18.01 WPM

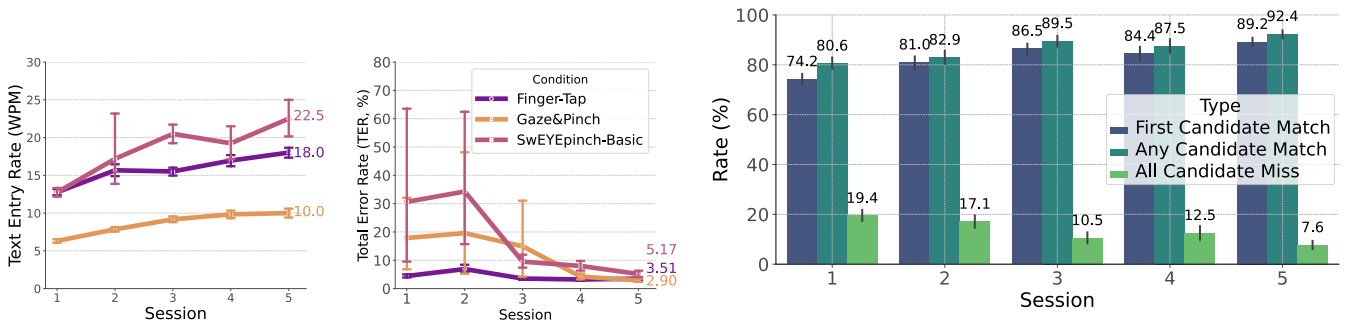


Figure 3: Performance results from US1 across five sessions. From left to right, WPM for the three techniques in US1, TER by the conditions in US1. *SwEYEpinch-Basic* error and match rates across sessions. Each bar represents the mean percentage of text-entry outcomes for three categories: (1) *First Candidate Match*: the first suggestion was correct; most desirable because the user can move on to the next word if the first candidate matches, (2) *Any Candidate Match*: at least one suggestion was correct, and (3) *All Candidate Miss*: no suggested candidates matched. Overall, the figure reveals how SwEYEpinch-Basic's suggestion accuracy evolves with practice over multiple sessions. Error bars show standard errors.

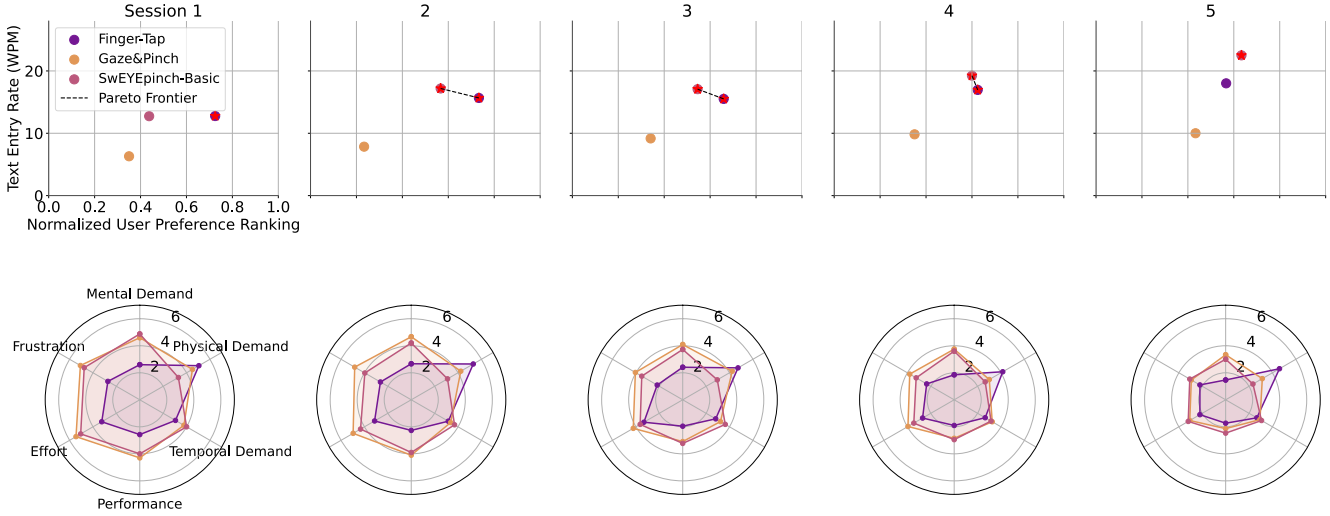


Figure 4: Top: Pareto frontiers showing normalized user preference vs. WPM. Bottom: Raw NASA TLX scores.

Condition	Learning Rate (WPM/Session)
SwEYEpinch-Basic	2.10
Gaze&Pinch	1.07
Finger-Tap	1.42

Table 1: Average learning rate (WPM per session).

($p = 2.55 \times 10^{-5}$) and *Gaze&Pinch* 10.01 WPM ($p < 10^{-32}$). Participants linked the gain to forgiving decoding and reduced letter-by-letter micromanagement: P104 "*Swipe was easy to understand and allowed for errors with the suggestion box*," P118 "*Swiping did incredibly well at guessing the words I was trying to type.*"

SwEYEpinch-Basic does not meet accuracy non-inferiority at Session 5, not supporting H1.2. TER was higher than both baselines at Session 5: 5.17% vs. *Finger-Tap* 3.51% ($p = 0.004$) and vs. *Gaze&Pinch* 2.90% ($p = 1.65 \times 10^{-4}$); *Finger-Tap* and *Gaze&Pinch* did not differ ($p = 0.088$). Without knowing what word will be predicted during a swipe, users point out they have to frequently gaze upward and check: P134: "*I kept having to look at the input field and it would ruin the word I was typing.*" Several participants found a workable tactic—anchoring the first letter—to stabilize recognition (P117: "*Focusing on getting the first letter correct really helped.*"). Because the decoder conditions on the pinch-onset location, it prunes the lexicon to candidates whose first letter lies near that start point. These observations directly motivate *SwEYEpinch*'s live mid-swipe

Table 2: Decoder comparison on US1 (19,328 traces; n=40). Running time (RT) or latency is the average time it takes to decode the candidates from the swipe trace. Our method runs DTW on I-VT+DBSCAN–pruned traces. †ST-DTW: spatiotemporal DTW. †† w/o td: without time dimension

Method	Top-1 (%)	Top-4 (%)	RT (ms)
SwEYEpinch (ST-DTW†)	77.9	87.5	2.38
SwEYEpinch (w/o td††)	72.1	85.9	2.32
EyeSwipe [24]	75.7	86.1	18.4
HGaze [11]	71.9	81.3	2.44

candidates and low-effort correction (mid-swipe deletion, deletion peek window) to reduce TER.

SwEYEpinch-Basic sits on the speed–preference Pareto frontier and dominates Gaze&Pinch, supporting H1.3. The Session 5 Pareto plot (WPM vs. preference) places *SwEYEpinch-Basic* on the frontier and dominating *Gaze&Pinch*; *Finger-Tap* trades familiarity for higher physical demand (Fig. 4). Comments reflect this trade-off: P108 "*FingerTap is reliable but very physically taxing; swipe was fun and easy to fix when it went wrong*," P112 "*Once I mastered the coordination, swipe was the simplest to use*."

Comparison to decoders in prior works. To contextualize our algorithmic design, we conducted a post-hoc evaluation on the gaze traces collected from US1's *SwEYEpinch-Basic* condition. Alongside our decoder with filtered gaze trace and spatiotemporal DTW, we re-implemented the two most relevant gaze-trace baselines: (i) *EyeSwipe*—DTW on (near-)raw gaze with simple outlier removal, and (ii) *HGaze*—Fréchet distance on a temporally averaged (smoothed) trace. As Table 2 shows, SwEYEpinch yields the best Top-1/Top-4 while remaining in the same latency class as DTW variants; EyeSwipe is slower because it aligns largely raw traces, whereas we prune with I-VT+DBSCAN. Removing timestamps in our DTW drops Top-1 by 5.8 points (77.9→72.1), highlighting the benefit of the weak forward-time prior; HGaze underperforms due to Fréchet's sensitivity to single noisy fixations. We did not include *GlanceWriter* in this offline comparison because its delimiter (crossing a line above the keyboard) changes the gaze-trace geometry, so replaying *SwEYEpinch* traces would be invalid. Instead, we compare against *GlanceWriter XR* directly in US2.

5.3.1 Hyperparameter sensitivity. Using the same US1 corpus as the decoder-baseline comparison, we swept the core preprocessing parameters: I-VT velocity threshold in $[50, 150]$ deg/s, DBSCAN $\varepsilon \in [0.05, 0.15]$ (normalized keyboard units), and $minPts \in [2, 6]$. Across this grid, mean per-trace latency was essentially unchanged (2.32–2.47 ms; mean 2.39 ms). Accuracy improved when lowering the I-VT threshold from 150 to 100 deg/s and then saturated (<1% additional gain). Varying DBSCAN ε or *minPts* had a negligible effect on accuracy. These results support the defaults used in our main experiments and indicate that the decoder is robust to reasonable parameter choices.

Design insight. These results reinforce our principle to *decouple targeting from commitment*: gaze for fast, low-effort tracing and a tiny pinch to commit yields substantial speed gains, with a small accuracy cost that, as later studies would reveal, can be mitigated via mid-swipe feedback and low-effort correction tools.

6 USER STUDY 2: SWEYEPINCH AND OTHER GAZE-SWIPE BASELINES

US1 confirmed our design intuition "gaze for targeting, pinch to confirm" is advantageous: pinch-delimited gaze swipe (*SwEYEpinch-Basic*) can beat production-style, letter-by-letter input on speed and preference. However, it also exposed a key limitation: candidates appear only at commit, prompting off-keyboard verification glances that contribute to higher error rates. We therefore developed an improved version, which we describe next.

6.1 Improving SwEYEpinch-Basic: SwEYEpinch

Participants in US1 found SwEYEpinch-Basic to be the most preferred text-entry method overall, even though the speed advantage emerged later as the sessions progressed. To further enhance both the efficiency and usability of SwEYEpinch-Basic, we introduce a refined variant called *SwEYEpinch*.

SwEYEpinch introduces two key improvements over SwEYEpinch-Basic: **(1) Mid-Swipe Prediction**, which displays word candidates as the user swipes, and **(2) Mid-Swipe Deletion**, which allows users to cancel an in-progress swipe if the predictions are incorrect. We describe each in detail below.

Mid-Swipe Prediction. Inspired by the Swype keyboard, we display candidate words above the current key whenever the user saccades from one letter to another *within* the same swipe. This preview allows the user to see potential completions without looking at every letter of the intended word. However, unlike touch-anchored Swype, when a user has just started an eye swipe, and relatively few gaze points have been captured, the gaze-based distance matching can be unreliable. It often proves more effective to rely on the language model.

To accommodate this, we multiply the fusion parameter α (see Section 3.2) with an adaptive weight that decreases linearly from 3 to 1 as the number of gaze points increases from 1 to 3. By doing so, we triple the importance of the n-gram probabilities when there is only one gaze point available, then smoothly decrease to the original α when there are three or more gaze points. This increases the influence of the language model when evidence is sparse and smoothly returns to the base α once the trace stabilizes. Our choices for these values were informed by pilot studies.

While mid-swipe prediction is standard on mobile touchscreens (e.g., Android), porting this interaction to gaze is non-trivial due to signal disparity. Touch input is relatively sparse and precise (high signal-to-noise ratio). In contrast, gaze data is high-frequency (200 Hz in our setup), voluminous, and prone to jitter and microsaccades. Naively applying standard decoding (e.g., frame-by-frame DTW on raw gaze) introduces latency that breaks the feedback loop. Prior work on gaze-swipe systems [9, 11, 22, 24, 56] delays prediction until after a swipe is complete. Therefore, our primary technical contribution is the **optimization pipeline** that makes continuous gaze-decoding feasible. By chaining velocity-thresholding (I-VT) with density-based clustering (DBSCAN), we reduce the swipe trace

cardinality by over 90% (see Appendix A), allowing our Spatiotemporal DTW to run within the required frame budget for fluid UI updates.

Mid-Swipe Deletion. In early testing, we observed that users sometimes recognized *during* a swipe that the emerging candidate words were incorrect, and they could not make a correction mid-gesture. This is especially true when the starting letter is wrong because we filter the candidates by gaze position at the start of a swipe. With *SwEYEpinch-Basic*, they would have to abort the swipe by releasing the pinch, then fixate and pinch again on the delete key. Instead, *SwEYEpinch* allows the user to *cancel* a swipe mid-gesture: if none of the displayed mid-swipe predictions are correct, the user can continue swiping to the delete key and release pinch there, discarding the swipe entirely. This "Mid-Swipe Deletion" streamlines error correction by eliminating a separate "exit + delete" action.

Deletion Peek Window. During our early testing, we observed that the user, when needing to delete longer text, often had to repeatedly shift their gaze up and down between the delete key to pinch at it and the text input field to check their inputs. To mitigate this disruptive behavior, we added a small contextual window that appears above the deletion button when the user gazes at it; this peek window displays the content of the input box. This way, the user does not need to shift their gaze back to the actual input box to check its content while deleting.

6.2 Other XR-native, Gaze-Swipe baselines

Moreover, to isolate specific contributions of our design against comparable *XR-native* techniques (i.e., those not requiring a physical surface like TagSwipe [22]), we benchmark against two representative gaze-swipe approaches: *SkiMR* and *GlanceWriter XR*.

SkiMR [16]. This method introduces a dwell-free gaze swipe where users confirm word entry by explicitly fixating on the *space* key instead of relying on dwell timers or pinch gestures. Apart from the explicit space-bar confirmation and decoding algorithm, all other aspects of the method align with standard gaze-swipe approaches.

GlanceWriter XR. We adapt the original desktop GlanceWriter [9] to XR. Users start and stop a swipe by entering and exiting the keyboard area.

We chose *SkiMR* and *GlanceWriter XR* because we restrict US2 to XR-native, gaze-swipe techniques that satisfy the same HWD-only sensing constraints as SwEYEpinch. This allows us to isolate the effect of the delimiter (pinch vs. space-key fixation vs. line crossing) and mid-swipe prediction without introducing confounds from additional hardware (e.g., touchscreens) or modalities (e.g., voice).

These baselines do not involve hand motion but still couple targeting and commitment to gaze alone. US2 therefore asks whether bringing *mid-swipe prediction* into a pinch-delimited design can cut verification glances and raise entry rate without harming accuracy—and how much of the gain comes from the *delimiter* itself. After limited practice (three sessions), we test three hypotheses:

- **H2.1.** *SwEYEpinch* (mid-swipe prediction) is faster than *SwEYEpinch-Basic* (post-swipe only) with equal or lower TER.
- **H2.2.** Pinch-delimited swipes outperform gaze-only methods (*SkiMR*, *GlanceWriter XR*) in WPM and preference without an accuracy penalty.
- **H2.3.** The two-stage pruning + spatiotemporal DTW decoder (`Gaze2Word`, Algorithm 1) yields higher top-1/any candidate-match rates than *GlanceWriter XR*'s prefix-trie decoder (under the same apparatus/keyboard), translating to higher WPM without a TER increase.

6.3 Study Design

The procedure is identical to that of US1, except that participants take part in three sessions instead of five.

6.4 Participants

Twenty-one participants (ages 19–34; $\bar{x} = 23.6$; 8 male, 13 female) completed US2. All reported proficient English and normal or corrected-to-normal vision; one was left-handed and one reported a history of strabismus. XR familiarity was common (66.7%), but only one participant had prior XR-typing experience.

6.5 Results and Discussion

Mid-swipe preview boosts speed without hurting accuracy, supporting H2.1. *SwEYEpinch* was faster than *SwEYEpinch-Basic* in every session (S1: 16.56±8.44 vs. 12.40±7.05 WPM; S2: 18.78±8.86 vs. 14.55±7.09; S3: 22.25±8.83 vs. 16.64±7.30; all $p<0.001$; Fig. 6a). Learning rates echoed this advantage (2.85 vs. 2.10 WPM/session; Table 3). TER did not worsen with previews and was lower in S2 (8.96% vs. 13.59%, $p=0.013$), becoming indistinguishable by S3 (n.s.; Fig. 6b). Participants attributed the gain to fewer "finish–inspect–fix" cycles: `P204` (S2) "*I could stop as soon as my word showed up.*" Together, these results **support H2.1**.

Pinch beats gaze-only delimiters on rate with no accuracy penalty, supporting H2.2. Pinch-delimited swipes (*SwEYEpinch*/*SwEYEpinch-Basic*) outpaced both gaze-only baselines across sessions (vs. *GlanceWriter XR* and *SkiMR*: all $p<0.001$ except *SwEYEpinch-Basic* vs. *GlanceWriter XR* in S1 $p=0.014$ and S3 $p=0.008$; Fig. 6a), consistent with avoiding extra saccades/dwell costs. TER showed no pinch penalty; by S3 both pinch conditions were lower than *SkiMR* (6.48% and 8.05% vs. 16.65%, both $p<0.001$), with other pairwise TERs not significant (Fig. 6b). Participants emphasized the control of an explicit, low-effort delimiter: `P221` "*It was easier to decide exactly when the word ended.*" These results **support H2.2**.

Decoder-level evidence matches the speedup: by S3, we yield more top-1/any matches, supporting H2.3. By S3, the *SwEYEpinch* decoder surpassed *GlanceWriter XR*'s prefix-trie on key outcomes (Fig. 6c): *First Candidate Match* 87.5% vs. 81.6% ($p=1.1\times10^{-4}$) and *Any Candidate Match* 89.5% vs. 85.4% ($p=0.003$); *All Candidate Miss* trended lower (10.5% vs. 12.4%, n.s.). Although prefix-trie led slightly in S1 (top-1 79.2% vs. 76.1%, $p=0.028$), this reversed as users adapted to pinch-delimited swipes and mid-swipe decoding, reducing verification and correction overhead. One participant captured this learnability: `P215` (S3) "*Auto-suggestions came*

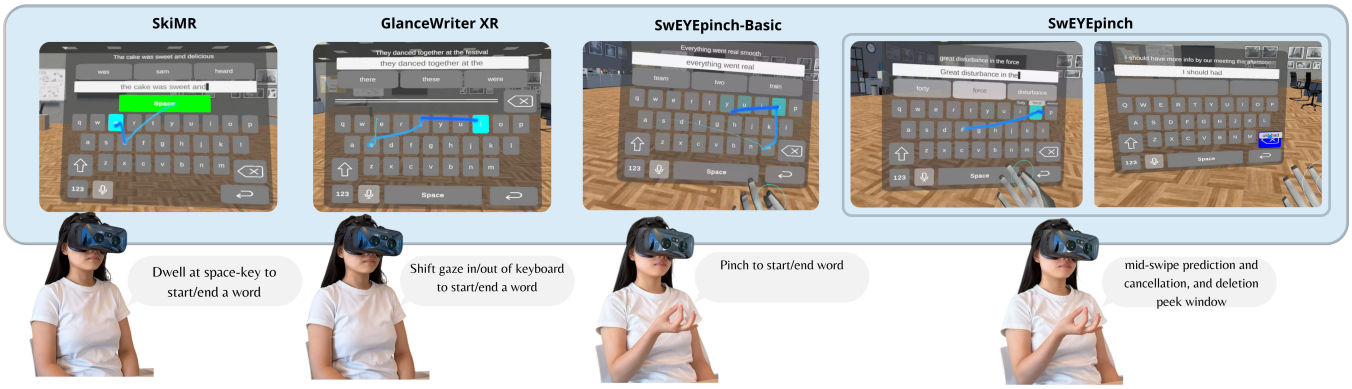


Figure 5: Techniques evaluated in US2 are gaze-swipe baselines with different delimiters: *SkiMR* (the delimiting space key is placed on top of the keyboard following [16]), *GlanceWriter XR*, and our proposed *SwEYEpinch-Basic* and its improved version: *SwEYEpinch*.

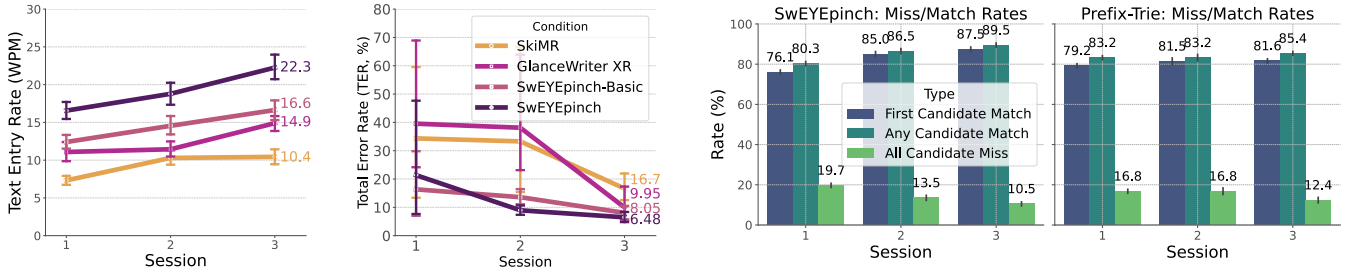


Figure 6: Performance results from US2. From left to right: WPM for the three techniques across sessions, TER across conditions. Error bars show standard errors; the match and miss rates are explained in Figure 3c. Comparing the decoding algorithm: *SwEYEpinch* vs GlanceWriter [9].

much quicker as I got used to moving with my eyes." These results **support H2.3**.

Condition	Learning Rate (WPM/Session)
SkiMR	1.49
GlanceWriter XR	2.05
SwEYEpinch-Basic	2.10
SwEYEpinch	2.85

Table 3: Average learning rate (WPM per session).

Learning trajectories. Design Insight. US2 isolates two levers that matter: *mid-swipe preview* (faster with equal or lower errors) and the *pinch delimiter* (faster than gaze-only without an accuracy cost). Qualitative accounts match this picture: previews curb verification glances and shorten swipes, and pinch provides a precise, low-effort "when" that many participants preferred (P413: "*No-hands methods took more concentration than the pinch methods.*").

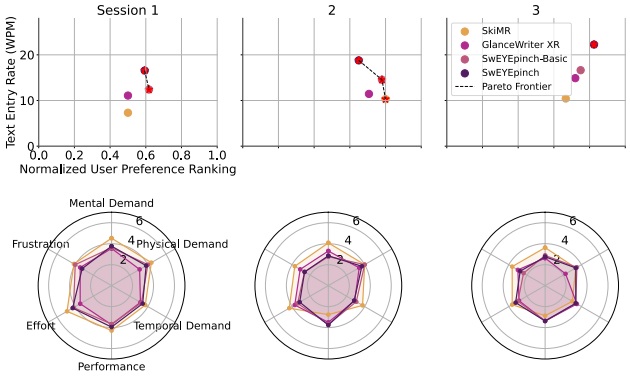


Figure 7: Top: Pareto frontiers showing normalized user preference vs. WPM. Bottom: Raw NASA TLX scores, across the three sessions in US2

7 USER STUDY 3: BENCHMARKING AGAINST STRONGER BASELINES

So far, US1 established that a minimal hybrid (*SwEYEpinch-Basic*)—using gaze for *where* and a tiny pinch for *when*—sits on the speed–preference Pareto frontier versus production-style, letter-by-letter input. US2 then showed *why* the hybrid works: mid-swipe previews cut verification glances, and the pinch delimiter avoids gaze-only activation costs.

US3 tests *external validity and skill transfer*. We ask whether the design principle *decouple targeting from commitment*, plus mid-swipe feedback, still wins against production-realistic contenders. To conduct a thorough comparison, we selected two competitors that address the algorithmic and modality limitations of previous baselines:

Finger-Tap with Prediction&Completion. To address the algorithmic disparity in US1 where word-level swiping competed against character-level tapping, we introduce this enhanced baseline. It extends *Finger-Tap* with *word completions* and *next-word prediction*, similar to smartphone keyboards. While such predictive features are standard on mobile devices, they are often omitted in XR text entry studies (e.g., DwellType, Controller-Tap). Including these predictive features brings comparison baselines to the same predictive power used in swipe methods. The *mid-swipe deletion* feature from *SwEYEpinch* is also added here.

Hand-Swipe. To test the modality difference (Eye vs. Hand) using identical decoders, we implemented *Hand-Swipe* as a *hand-pointing* analogue of *SwEYEpinch*: the user initiates a swipe by pinching, then releases the pinch to finish the swipe. This technique is inspired by *Vulture* [36], a seminal mid-air word-gesture keyboard that utilizes a ray-cast from the hand and a pinch delimiter. Vulture is well established in the literature [8, 10, 15] and has demonstrated a high entry rate (up to 28 WPM), ensuring that our baseline represents a high-performance standard for mid-air gestures rather than a naive implementation. Apart from substituting gaze with hand movement, all other aspects (e.g., mid-swipe candidate previews and cancellation) match those of *SwEYEpinch*.

Finally, US3 includes two cohorts: participants new to our system and *US1 alumni who used only SwEYEpinch-Basic*. This lets us test whether skills learned with the basic hybrid transfer to *SwEYEpinch*, but *do not* transfer to methods that break the unique *where*/*when* mapping.

- **H3.1.** After limited practice, *SwEYEpinch* achieves higher WPM than *Finger-Tap w/ Prediction&Completion* and *Hand-Swipe*.
- **H3.2.** *SwEYEpinch* attains TER that is no worse than the contenders (and *lower* than *Hand-Swipe*), demonstrating that speed gains do not come at an accuracy cost.
- **H3.3.** Users improve in performance the fastest with *SwEYEpinch*, reflecting reduced verification/correction overhead from mid-swipe previews&cancellation, and deletion peek window.
- **H3.4.** *SwEYEpinch* lies on (or dominates) the speed–preference Pareto frontier and yields equal or lower raw NASA TLX scores than the contenders.
- **H3.5.** Relative to new participants, US1 alumni more rapidly exploit mid-swipe previews in *SwEYEpinch*: by Session 2 they produce more characters per swipe point (fewer swipe points per word) and shorter swipe paths, with faster time-to-confirm once the intended candidate first appears and higher top-1/any-match rates. No cohort advantage is expected for *Finger-Tap w/ Prediction&Completion* or *Hand-Swipe*.
- **H3.6.** Prior experience influences learning: the alumni's advantage (or slope) is larger for *SwEYEpinch* than for *Finger-Tap* or *Hand-Swipe*, consistent with transfer specific to the decoupled targeting/commitment design.

7.1 Study Design

The procedure is identical to US2.

7.2 Participants

Forty-one volunteers participated in this study (ages 21 to 41; $\bar{x}$ = 24.9; 20 male, 21 female). All participants reported having normal or corrected-to-normal vision and no issues with hand movement. To study the effect of continued learning, among the 41 participants, 28 of them had previously participated in US1, and the rest were new. All but one participant was right-handed. All participants were familiar with typing on both computer and smartphone keyboards and had prior exposure to XR systems. However, 7 of 13 (46.7%) US3 only participants had never used XR keyboards, while the rest reported only monthly or weekly usage. All participants were fluent in English.

7.3 Results and Discussion

	Learning Rate (WPM/Session)	
Condition	**US3-Only Users**	**US1 Users**
SwEYEpinch	5.09	2.50
Finger-Tap w/ PC†	1.44	1.63
Hand-Swipe	2.28	0.96

Table 4: Average learning rate (WPM per session) by condition for participants in US1 vs. US3. †Finger-Tap w/ PC = finger tapping with prediction&completion.

SwEYEpinch is the fastest against production-realistic baselines, supporting H3.1. By Session 3, *SwEYEpinch* reached 26.2±13.1 WPM and significantly outperformed *Finger-Tap w/ Pred.* (p<0.001) and *Hand-Swipe* (p<0.001); see Fig. 9a.

Absolute TER non-inferiority to Finger-Tap did not hold, but SwEYEpinch reduced errors the fastest, not supporting H3.2. By Session 3, *Finger-Tap w/ Pred.* had lower TER than *SwEYEpinch* (p=0.018), while the *SwEYEpinch* vs. *Hand-Swipe* TER difference was not significant (p=0.31; Fig. 9b); thus, **H3.2** is not supported on an absolute basis. However, *SwEYEpinch* showed the steepest TER decline and higher first-candidate match rates than *Hand-Swipe* across sessions ($p<10^{-4}$; Fig. 10), consistent with reduced correction overhead.

SwEYEpinch shows the largest WPM gains per session, supporting H3.3. Learning-rate estimates (Table 4) show *SwEYEpinch* improving most rapidly within each cohort (US3-only:

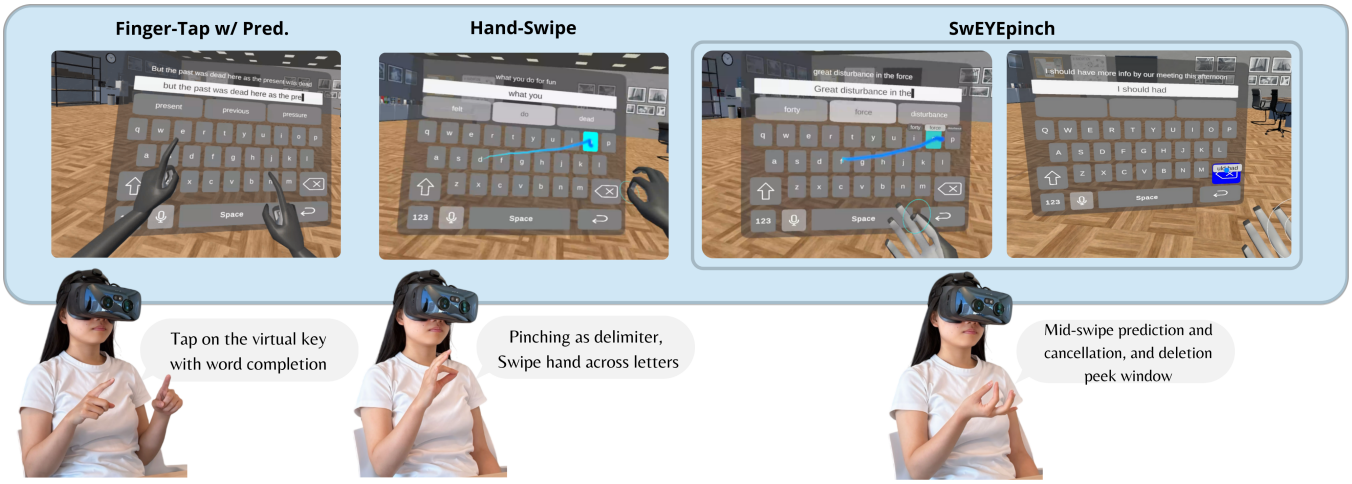


Figure 8: Techniques evaluated in US3: the production-realistic *Finger-Tap with word completion and next word prediction* (left), *Hand-Swipe* that shares the same features as *SwEYEpinch* but using the hand as the cursor, and *SwEYEpinch* from US2.

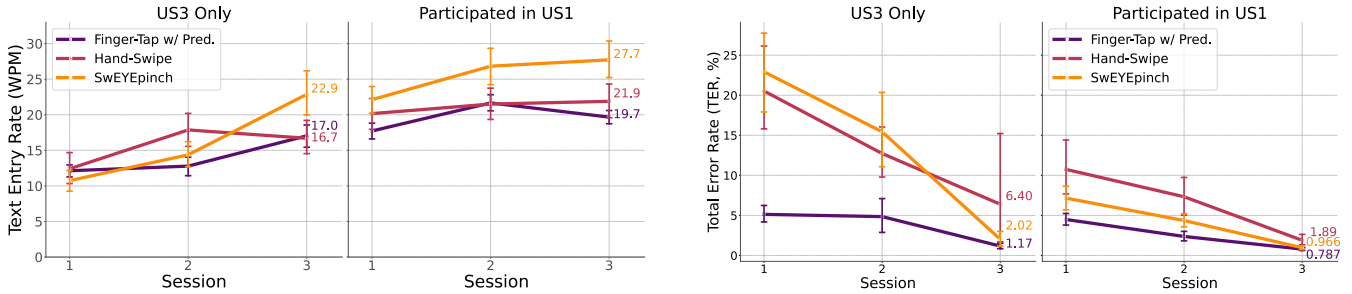


Figure 9: Performance results from US3. (Left 2) Average text entry speeds (in WPM) and (Right 2) TER by the conditions, separated for participants who only participated in US3 and those who also participated in US1. Error bars show standard errors.

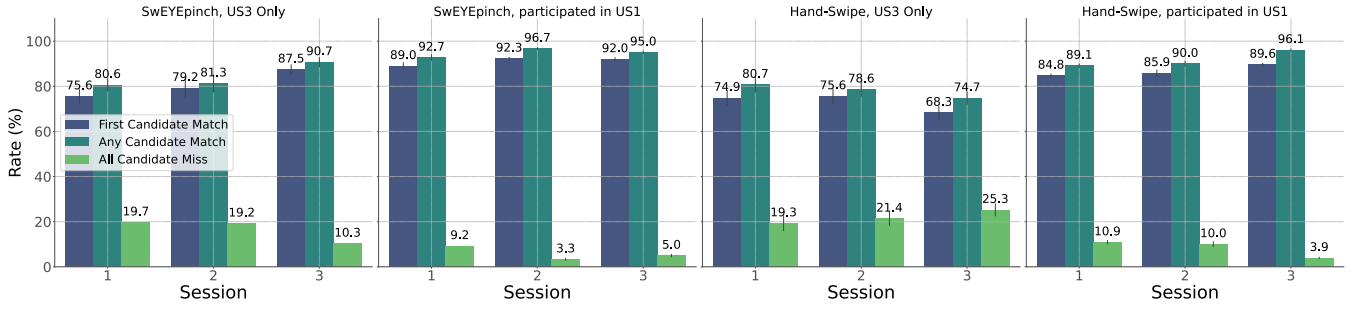


Figure 10: The match and miss rates are explained in Figure 3c. US3-only participants' miss rates do not improve in Hand-Swipe.

5.09 WPM/session; US1-alumni: 2.50 WPM/session), exceeding both *Finger-Tap w/ Pred.* and *Hand-Swipe*. This aligns with mid-swipe previews and low-effort cancellation reducing verification/correction time.

SwEYEpinch remains Pareto-optimal with competitive workload, supporting H3.4. By Session 3, *SwEYEpinch* lies on the speed–preference Pareto frontier for both participant groups (Fig. 11, top). NASA TLX decreases across sessions for all techniques, with *SwEYEpinch* ending at equal or lower workload than alternatives (bottom).

Prior exposure yields an early advantage; novices reach parity by Session 3, supporting H3.5. Alumni produced shorter

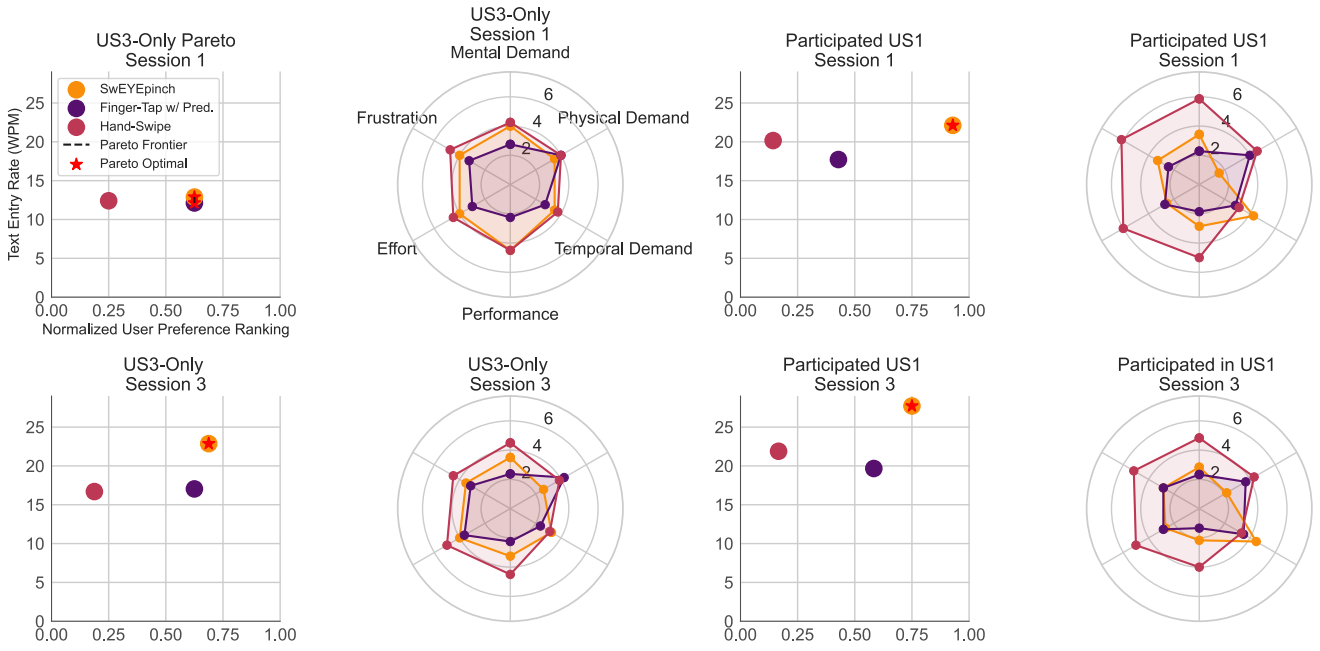
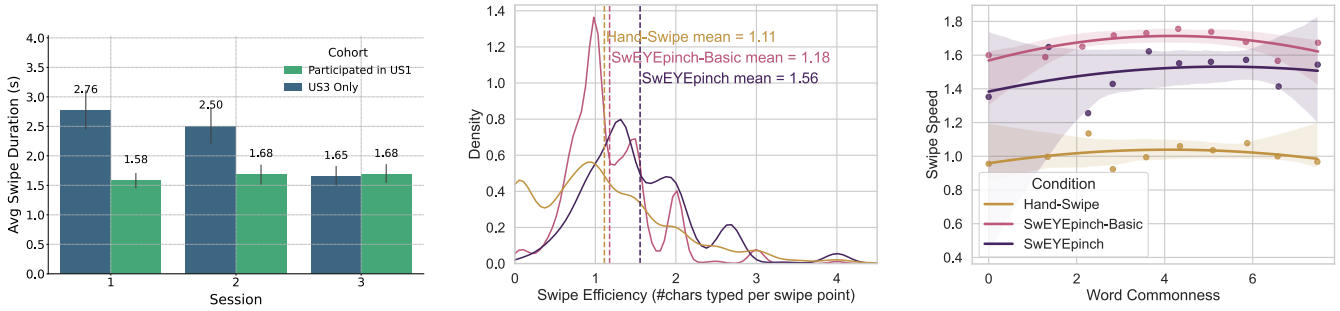


Figure 11: Pareto frontiers (normalized preference vs. WPM) and raw NASA TLX scores at Sessions 1 and 3 for two participant groups: those who only participated in US3 (left four figures) and those who also participated in US1 (right four figures).



(a) (b) (c)

Figure 12: Mechanism uptake and efficiency in US3. (a) Average *SwEYEpinch* swipe duration. (b) Swipe Efficiency distribution—plotted as a kernel density estimate (KDE) over the per-word effort metric. Higher values indicate fewer waypoints per letter. (c) Word commonness (Zipf frequency from [50]) versus swipe speed with quadratic fits (higher Zipf = more common words).

SwEYEpinch swipes in S1–S2 ($p \leq 0.031$) but converged with novices by S3 (p=0.90; Fig. 12,(a)). Across conditions, *SwEYEpinch* shows higher swipe efficiency than *SwEYEpinch-Basic* and *Hand-Swipe* (Fig. 12,(b)), consistent with preview-and-confirm behavior rather than full-word tracing.

Cohort×Technique effects reflect initial performance more than learning rate: alumni start faster on SwEYEpinch, but novices learn it faster, supporting H3.6. At Session 1 the cohort gap is largest for *SwEYEpinch*, smaller for *Hand-Swipe*, and smallest for *Finger-Tap with Prediction&Completion* (Fig. 9a), indicating technique-specific transfer. Slopes invert: novices ramp faster on *SwEYEpinch* (5.09 vs. 2.50 WPM/session), while differences are smaller for the baselines (Table 4). Thus, H3.6 is supported for *initial performance* but not for a larger alumni *slope*.

Design Insight. Mid-swipe previews plus a pinch delimiter promote efficient *partial* swipes and early commit, yielding top entry rate and fast learning while keeping workload competitive. Transfer benefits initial performance when the where/when mapping is preserved (*SwEYEpinch* vs. hand/tap baselines), and novices quickly

acquire the same preview-and-confirm strategy. In products, surface live suggestions early, make early-commit cheap and visible, and keep low-effort correction (e.g., peek/delete) to accelerate convergence.

8 USER STUDY 4: TOWARDS EVERYDAY TYPING SPEEDS—SEVEN DAYS OF SWEYEPINCH

US1–US3 established *why* the hybrid works (mid-swipe previews + pinch delimiter), *that* it beats production-style baselines, and *where* transfer appears. As a natural extension, *does SwEYEpinch keep improving with daily use and can it reach ordinary keyboard speeds?* To our knowledge, XR text-entry papers rarely put an interface through a longitudinal "test of time". US4 accomplishes a week-long, multiple-times-a-day study over 30 consecutive sessions that tracks individual learning curves against normal keyboard performance.

Guided by classic longitudinal typing work [37] and by the learning patterns observed in US1–US3, we test the following:

- **H4.1 (Sustained learning).** Across a week of daily use, *SwEYEpinch* shows a positive per-user trend in WPM and does not plateau mid-week.
- **H4.2 (Everyday-speed attainability).** With routine practice, *SwEYEpinch* approaches or crosses the lower band of everyday keyboard speeds for non-developer users.
- **H4.3 (Experience moderates, but everyone improves).** Prior exposure shapes the *shape* of the curve (refinement vs. rapid gains), yet all users realize meaningful improvement over time.

8.1 Study Design

Each participant was required to complete a total of 30 sessions, distributed across seven consecutive days, to balance exposure with rest and consolidation.

Each session consisted of 14 transcription tasks, where participants typed full sentences using *SwEYEpinch*. All participants were given the same sentences at the same session index. No phrases were repeated. This design resulted in 420 total transcription trials per participant (14 phrases × 30 sessions).

8.2 Participants

Recruitment in US4 was more limited than in other studies due to the substantial time commitment; nine participants completed all sessions. We sampled three experience cohorts (ages 20 to 31; $\bar{x}$ = 25.8 years), 7 male, 2 female:

(1) SwEYEpinch- and XR-novices. P403, P406, and P407 had no prior exposure to our techniques, had never typed in XR, and reported only minimal XR use.

(2) XR-experienced, SwEYEpinch-naïve. P405, P408, P409 had substantial prior XR experience (as regular users or developers) but had not used *SwEYEpinch* before US4.

(3) SwEYEpinch-experienced / experts. P401, P402, and P404 had prior exposure to *SwEYEpinch* through earlier studies or development work, and thus entered US4 with intermediate-to-expert proficiency.

8.3 Results and Discussion

SwEYEpinch keeps getting faster with daily use and does not plateau mid-week, supporting H4.1. All nine participants show positive overall learning slopes (0.77–1.65 WPM/session) and rising trajectories across 30 sessions (Fig. 13). Learning remains active well beyond Session 10 for both experts and non-experts: for example, P402 and P404 maintain strong gains in the 11–20 block (1.77 and 1.82 WPM/session), while novices such as P406 and P407 continue to accelerate late in the week (e.g., P407: 2.98 WPM/session in Sessions 21–30). These trends contradict a mid-week plateau. As one participant put it, "*By midweek I wasn't tracing whole words anymore—once my word appeared, I just pinched and moved on.*"

Everyday typing speeds are attainable with routine practice, partially supporting H4.2. Five participants reached or exceeded the 54 WPM band typical of everyday touch typing (P401: 60.6; P402: 64.7; P405: 56.5; P407: 57.0; P409: 60.9 median best), and the remaining four converged on high-40s (45.4, 45.9, 46.2, 48.9 WPM for P403, P406, P404, P408). These speeds emerged from repeated daily use rather than single-session tuning (Fig. 13). "*It went from 'cool demo' to 'I can actually type like this,'*" noted P402.

Experience shapes the curve's *shape*, but everyone improves meaningfully, supporting H4.3. Experts who entered with prior SwEYEpinch experience (P401, P402, P404) started high and then refined: P401's learning rate tapered over the three 10-session blocks (1.93 → 1.14 → 0.263 WPM/session), while P402 and P404 showed mid-study accelerations (peaks of 1.77 and 1.82 WPM/session). XR-experienced but SwEYEpinch-naïve users (e.g., P405) exhibited solid gains early on (1.53 WPM/session in Sessions 1–10), then consolidated. Novices (P403, P406, P407) began at much lower speeds but posted some of the largest overall gains, including late-week acceleration in some cases (e.g., P407's 2.98 WPM/session in Sessions 21–30). Across all participants, performance improvements were substantial (ΔWPM: 24.1–49.1), echoing US2–US3: once users come to trust mid-swipe previews, they shorten traces, commit earlier, and steadily push into everyday typing ranges.

Design Insights. US4 shows that *SwEYEpinch* has the potential for everyday viability through compounding efficiency: with repeated use, mid-swipe previews shift behavior from tracing to "see–confirm–move," producing shorter paths and steady WPM gains that persist beyond mid-week. Because the *where*/*when* mapping stays stable, skill transfers cleanly across days and experience levels, keeping progress monotonic. Reaching 60+ WPM for three participants and high-40s for the rest places *SwEYEpinch* inside the everyday keyboard envelope—the technique itself (previews + early commit) has the potential to be an interface people can live with.

9 DISCUSSION

This section synthesizes evidence from US1–US4 into design guidance for XR text entry. Our central result is that *decoupling targeting from commitment*—using gaze for *where* and a tiny explicit gesture for *when*—yields a better speed–effort trade-off than production-style baselines, especially when paired with *mid-swipe previews* and low-effort correction.

Principle: Decouple where *from* when. US1 showed that a minimal hybrid (*SwEYEpinch-Basic*) sits on the speed–preference Pareto

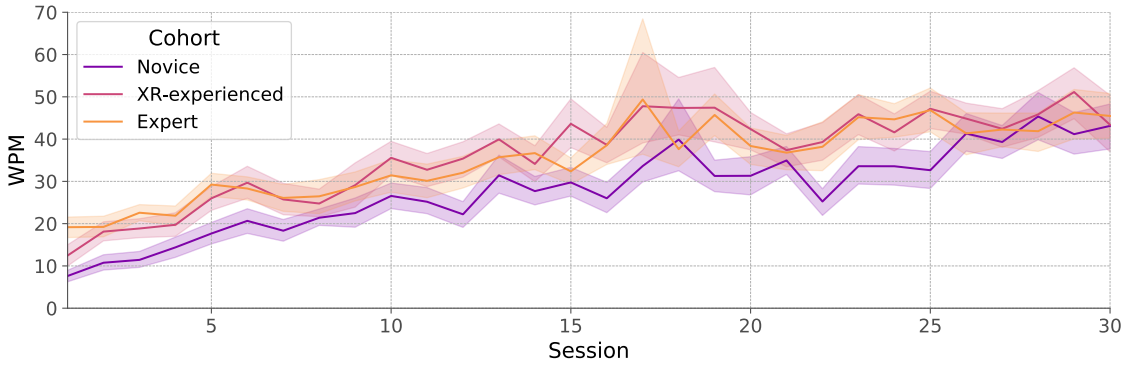


Figure 13: *SwEYEpinch* performance with extended daily sessions. Average WPM by participant cohort.

frontier versus *Finger-Tap* and *Gaze&Pinch* (Figure 3, Figure 4). US2 isolated *why*: the pinch delimiter outperformed gaze-only delimiters without an accuracy cost, and *mid-swipe previews* cut verification glances and raised WPM (Figure 6). US3 extended this against production-realistic contenders: *SwEYEpinch* was fastest while remaining Pareto-optimal on preference and competitive on workload (Figure 9, Figure 11). US4 demonstrated sustained gains, with three participants exceeding 60 WPM (Figure 13).

Mechanism: Previews enable partial swipes and early commit. Across studies, we see a consistent behavioral shift: users stop tracing once the intended word appears and commit with a small pinch. This shows up as shorter paths, fewer turns, and higher characters-per-swipe-point, alongside higher top-1/any candidate matches with *SwEYEpinch*'s decoder (Figure 6c, Figure 10). The qualitative geometry is visible in Figure 14: *SwEYEpinch* traces are more compact and often end mid-word, reflecting preview-and-confirm behavior rather than full-word tracing.

Design recommendations for XR text entry. **(R1) Keep the delimiter explicit and tiny.** A low-effort pinch provides precise *when* without the latency/ambiguity of dwell or line-crossing delimiters, improving entry rate with no accuracy penalty (US2; Figure 6). **(R2) Display live candidates near the gaze locus.** Mid-swipe previews reduce "finish–inspect–fix" loops and enable early commit, driving WPM gains (US2/US3; Figure 6, Figure 9). **(R3) Optimize the correction loop.** Make cancellation and deletion low-friction—retain in-gesture cancel to discard a bad swipe, whole-word rollback immediately after commit, and a *peek window* to minimize off-keyboard glances. **(R4) Bias decoding toward early partials.** Fuse a lightweight language model with a spatiotemporal matcher and ramp LM influence when evidence is sparse; this stabilizes early previews without harming later accuracy (US2; Figure 6c). **(R5) Minimize eye travel.** Keep candidates and critical controls on or just above the keyboard plane to avoid costly verification glances and preserve an "eyes-on-keyboard" loop. **(R6) Provide short-string safeguards.** Very brief traces are easy to over-commit; lightweight disambiguation cues (e.g., transient alternatives or stricter commit thresholds for very short paths) preserve speed without adding dwell.

Learning and transfer. When the *where*/*when* mapping is preserved, skills transfer: alumni started *SwEYEpinch* faster in US3, while novices quickly caught up by adopting preview-and-confirm strategies (shorter durations, higher efficiency; Figure 12). Product implication: onboarding can focus on the single mental model "trace until your word appears, then pinch," rather than mechanics training.

Everyday viability. US4's week-long regimen shows compounding efficiency: as users transition from tracing to "see–confirm–move," WPM continues to rise beyond mid-week, with two users surpassing 60 WPM and others approaching high 40s (Figure 13). For XR platforms, this suggests that a hybrid gaze–pinch design with live previews and low-effort correction is not just learnable—it can reach everyday speeds with routine use. These speeds are comparable to, or higher than, those reported for prior multimodal techniques that rely on touch surfaces or speech ([14, 22, 56]), while SwEYEpinch remains HWD-only and silent. This underscores why our in-study baselines focus on commercial XR techniques and HWD-only gaze/gesture methods: they reflect the practical comparison set for the everyday scenarios we target.

Overall, *SwEYEpinch* reframes gaze from a *click surrogate* to a *continuous intent signal*, with a tiny explicit delimiter for commitment. This pairing reshapes the speed–effort frontier relative to current XR baselines, and our studies outline the concrete UI/decoder choices that make it work in practice.

10 LIMITATIONS AND FUTURE WORK

Our investigation brings out several interactions, hardware, and ergonomic constraints. First, deletion is coarse: whole-word rollback is only available immediately after a commit; at other times, users must backspace character by character. More flexible mechanisms—e.g., gesture hotkeys [48], auto-repeat on hold, or layouts beyond QWERTY [34] with dedicated keys for word/character delete—could reduce correction burden. Second, tracking and actuation are not uniformly inclusive: infrared eye tracking and hand tracking can underperform for some users, and prolonged use can lead to eye fatigue. Future work should pursue setups that are robust to diverse user groups.

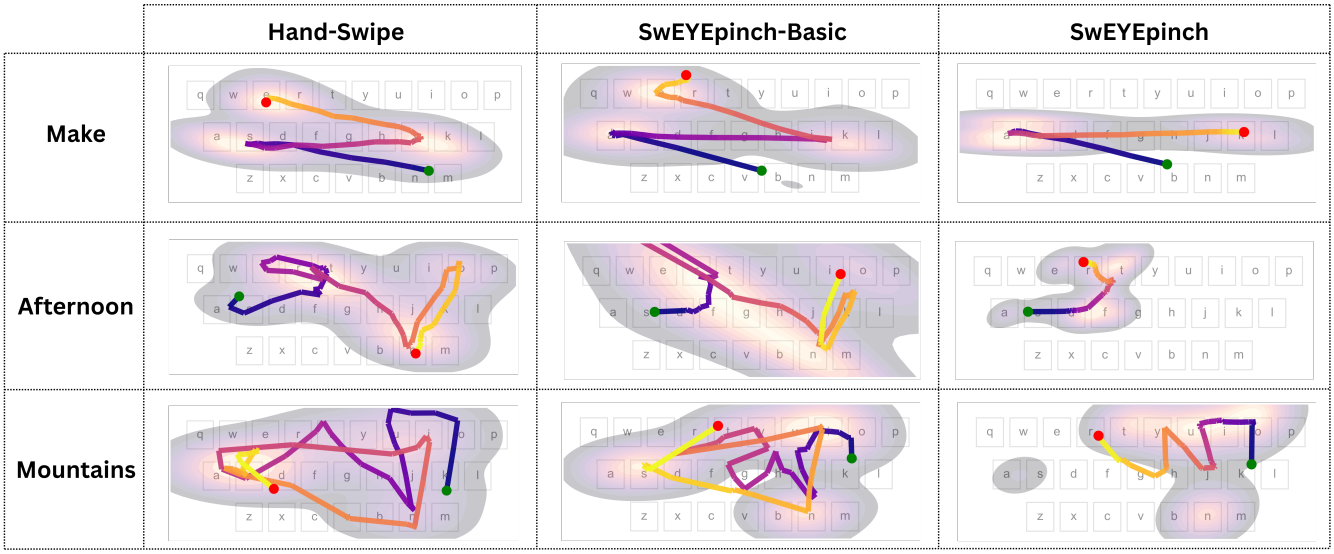


Figure 14: Average swipe paths for words ("make", "afternoon", "mountains") from three swipe-based techniques: *Hand-Swipe*, *SwEYEpinch-Basic*, and *SwEYEpinch*. The red and green dots indicate the average start and end position of the swipe traces. Compared to *Hand-Swipe* and *SwEYEpinch-Basic*, *SwEYEpinch* produces shorter paths with fewer turns and tends to end mid-word thanks to predictive completion.

In line with prior gaze–swipe text-entry work [9, 22, 56], our evaluation focused on speed, accuracy, and subjective workload rather than cybersickness or spatial-awareness outcomes. Our sessions were seated, and participants did not spontaneously report nausea or disorientation. Nonetheless, longer-term and mobile use (e.g., walking or commuting) may induce different responses. As our studies primarily report performance (WPM, TER) and preference, we cannot conclude how fatigue accumulates over prolonged use beyond the session durations studied. Future work should pair SwEYEpinch with standardized cybersickness measures (e.g., SSQ) and dual-task paradigms that stress navigation or spatial awareness, to more fully characterize how gaze–pinch text entry integrates into everyday XR scenarios.

Methodologically, our studies used transcription tasks that may under-represent real composition and dialogue. Naturalistic scenarios such as open-ended writing, and stratification by language proficiency would better capture cognitive load and vocabulary effects. Moreover, our in-study comparisons do not include techniques that use an external button/controller (e.g., GestureType [56]) or a touch surface (e.g., TagSwipe [22]). As a result, our conclusions are strongest for *HWD-only, silent, surface-free* XR typing, and we treat controller/surface-based techniques as external reference points rather than direct in-study baselines. Prior work suggests that sustained, targeted saccades and continuous visual focus can be fatiguing, and susceptibility to discomfort may increase with age. Our participant pools skewed young, so an important next step is to evaluate *SwEYEpinch* with older adults and to explicitly quantify fatigue/comfort alongside objective proxies (gaze travel distance, saccade counts, blink rate) during longer free-form typing.

On the systems side, our Python decoder leaves performance headroom; moving critical paths to accelerated kernels and on-device inference could further cut latency. Finally, decoding should adapt to behavior: early in a swipe, weight language priors more heavily; detect hesitation or ambiguous short traces to raise commit thresholds or surface targeted disambiguation—balancing speed and accuracy in situ.

11 CONCLUSIONS

This paper tested a simple idea for XR text entry: separate *targeting* from *commitment*—use gaze for the fast, low-effort *where* and a small explicit gesture for the *when*. Across four studies, this principle—instantiated as *SwEYEpinch*—improved the speed–effort balance without sacrificing accuracy. US1 placed a minimal hybrid on the Pareto frontier against *Finger-Tap* and *Gaze&Pinch*; US2 showed that a pinch delimiter outperforms gaze-only delimiters and that *mid-swipe previews* raise text entry rate without an accuracy cost; US3 confirmed wins over production-realistic contenders; and US4 demonstrated sustained learning to everyday speeds, with three participants surpassing 60 WPM.

Mechanistically, mid-swipe candidates, low-friction correction (in-gesture cancel, immediate rollback, deletion peek), and a low-latency spatiotemporal decoder with an *n*-gram prior enable *partial* swipes and *early commit*: shorter paths, fewer turns, higher first-candidate matches. For designers: keep the delimiter explicit and tiny; surface candidates in place while swiping; make correction effortless; and weight language priors early, handing off to spatial matching as evidence accrues. With code and data released (Section 3.2.1), we offer a replicable pattern for XR text entry that is fast, learnable, and ready to use beyond the lab.

ACKNOWLEDGMENTS

We thank our colleagues at Columbia University, especially Haoyan Chen, for assistance with figure preparation. We also thank Dr. Brian Smith for thoughtful discussions that informed this work. This research was supported in part by the Air Force Office of Scientific Research (FA9550-22-10337), the Army Research Laboratory (W911NF-19-2-0139, W911NF-19-2-0135, W911NF-21-2-0125), and the U.S. Department of Defense (N00014-20-1-2027).

ACKNOWLEDGMENT OF AI USE

We acknowledge the use of Generative AI tools to improve the grammar, style, and readability of this manuscript, but played no role in the data analysis, interpretation, or generation of the core findings presented.

REFERENCES

[1] Jiban Adhikary and Keith Vertanen. 2021. Text entry in virtual environments using speech and a midair keyboard. *IEEE Transactions on Visualization and Computer Graphics* 27, 5 (2021), 2648–2658.

[2] Mehmet Akhoroz and Caglar Yildirim. 2024. Poke Typing: Effects of Hand-Tracking Input and Key Representation on Mid-Air Text Entry Performance in Virtual Reality. In *Proceedings of the 26th International Conference on Multimodal Interaction*. 293–301.

[3] Richard Andersson, Linnea Larsson, Kenneth Holmqvist, Martin Stridh, and Marcus Nyström. 2017. One algorithm to rule them all? An evaluation and discussion of ten eye movement event-detection algorithms. *Behavior research methods* 49 (2017), 616–637.

[4] Tanya Bafna, Per Bækgaard, and John Paulin Hansen. 2021. Mental fatigue prediction during eye-typing. *PLOS One* 16, 2 (2021), e0246739.

[5] Arpit Bhatia, Moaaz Hudhud Mughrabi, Diar Abdlkarim, Massimiliano Di Luca, Mar Gonzalez-Franco, Karan Ahuja, and Hasti Seifi. 2025. Text Entry for XR Trove (TEXT): Collecting and Analyzing Techniques for Text Input in XR. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 1223, 18 pages. DOI: http://dx.doi.org/10.1145/3706598.3713382

[6] Costas Boletsis and Stian Kongsvik. 2019. Controller-based text-input techniques for virtual reality: An empirical comparison. *International Journal of Virtual Reality (IJVR)* 19, 3 (2019). DOI: http://dx.doi.org/10.20870/IJVR.2019.19.3.2917

[7] Gavin Buckingham. 2021. Hand tracking for immersive virtual reality: opportunities and challenges. *Frontiers in Virtual Reality* 2 (2021), 728461.

[8] 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.

[9] Wenzhe Cui, Rui Liu, Zhi Li, Yifan Wang, Andrew Wang, Xia Zhao, Sina Rashidian, Furqan Baig, I.V. 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, 1–13. DOI: http://dx.doi.org/10.1145/3544548.3581269

[10] 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.

[11] Wenxin Feng, Jiangnan Zou, Andrew Kurauchi, Carlos H Morimoto, and Margrit Betke. 2021. HGaze Typing: Head-gesture assisted gaze typing. In *ACM Symposium on Eye Tracking Research and Applications*. 1–11.

[12] 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.

[13] 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.

[14] Ramin Hedeshy, Chandan Kumar, Raphael Menges, and Steffen Staab. 2021. Hummer: Text entry by gaze and hum. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–11.

[15] Jay Henderson, Jessy Ceha, and Edward Lank. 2020. STAT: Subtle typing around the thigh for head-mounted displays. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–11.

[16] Jinghui Hu, John J Dudley, and Per Ola Kristensson. 2024. SkiMR: Dwell-free eye typing in mixed reality. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 439–449.

[17] Jinghui Hu, John J Dudley, and Per Ola Kristensson. 2025. Seeing and Touching the Air: Unraveling Eye-Hand Coordination in Mid-Air Gesture Typing for Mixed Reality. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 1222, 15 pages. DOI: http://dx.doi.org/10.1145/3706598.3713743

[18] Robert J. K. Jacob. 1991. The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Trans. Inf. Syst.* 9, 2 (April 1991), 152–169. DOI: http://dx.doi.org/10.1145/123078.128728

[19] Florian Kern, Florian Niebling, and Marc Erich Latoschik. 2023. Text input for non-stationary XR workspaces: Investigating tap and word-gesture keyboards in virtual and augmented reality. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (2023), 2658–2669.

[20] Jinwook Kim, Sangmin Park, Qiushi Zhou, Mar Gonzalez-Franco, Jeongmi Lee, and Ken Pfeuffer. 2025. PinchCatcher: Enabling Multi-selection for Gaze+Pinch. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 853, 16 pages. DOI: http://dx.doi.org/10.1145/3706598.3713530

[21] 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.

[22] Chandan Kumar, Ramin Hedeshy, I Scott MacKenzie, and Steffen Staab. 2020. Tagswipe: Touch assisted gaze swipe for text entry. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[23] Manu Kumar, Jeff Klingner, Rohan Puranik, Terry Winograd, and Andreas Paepcke. 2008. Improving the accuracy of gaze input for interaction. In *Proceedings of the 2008 symposium on Eye tracking research & applications*. 65–68.

[24] Andrew Kurauchi, Wenxin Feng, Ajjen Joshi, Carlos Morimoto, and Margrit Betke. 2016. EyeSwipe: Dwell-free Text Entry Using Gaze Paths. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 1952–1956. DOI: http://dx.doi.org/10.1145/2858036.2858335

[25] Ziheng 'Leo' Li, Haowen Wei, Ziwen Xie, Yunxiang Peng, June Pyo Suh, Steven Feiner, Paul Sajda, and others. 2024. Physiolabxr: A python platform for real-time, multi-modal, brain–computer interfaces and extended reality experiments. *Journal of Open Source Software* 9, 93 (2024), 5854.

[26] Xi Liu, Bingliang Hu, Yang Si, and Quan Wang. 2024. The role of eye movement signals in non-invasive brain-computer interface typing system. *Medical & Biological Engineering & Computing* 62, 7 (2024), 1981–1990.

[27] 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 (OzCHI '15)*. Association for Computing Machinery, New York, NY, USA, 192–196. DOI: http://dx.doi.org/10.1145/2838739.2838804

[28] Xueshi Lu, Difeng Yu, Hai-Ning Liang, and Jorge Goncalves. 2021. iText: Hands-free Text Entry on an Imaginary Keyboard for Augmented Reality Systems. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21)*. Association for Computing Machinery, New York, NY, USA, 815–825. DOI: http://dx.doi.org/10.1145/3472749.3474788

[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 *Proceedings of the 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 344–349.

[30] Tiffany Luong, Yi Fei Cheng, Max Möbus, Andreas Fender, and Christian Holz. 2023. Controllers or bare hands? a controlled evaluation of input techniques on interaction performance and exertion in virtual reality. *IEEE Transactions on Visualization and Computer Graphics* 29, 11 (2023), 4633–4643.

[31] Mathias N. Lystbæk, Ken Pfeuffer, Jens Emil Grønbæk, and Hans Gellersen. 2022. Exploring Gaze for Assisting Freehand Selection-based Text Entry in AR. *Proceedings of the ACM on Human-Computer Interaction* 6, ETRA (2022), 141:1–141:16. DOI: http://dx.doi.org/10.1145/3530882

[32] 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.

[33] I Scott MacKenzie and Kumiko Tanaka-Ishii. 2010. *Text entry systems: Mobility, accessibility, universality*. Elsevier.

[34] Päivi Majaranta and Kari-Jouko Räihä. 2002. Twenty years of eye typing: systems and design issues. In *Proceedings of the 2002 Symposium on Eye Tracking Research & Applications (ETRA '02)*. Association for Computing Machinery, New York, NY, USA, 15–22. DOI: http://dx.doi.org/10.1145/507072.507076

[35] Adam Mansour and Jason Orlosky. 2023. Approximated Match Swiping: Exploring more Ergonomic Gaze-based Text Input for XR. In *2023 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. 141–145. DOI: http://dx.doi.org/10.1109/ISMAR-Adjunct60411.2023.00037

[36] 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.

[37] Edgar Matias, I Scott MacKenzie, and William Buxton. 1996. One-handed touch typing on a QWERTY keyboard. *Human-Computer Interaction* 11, 1 (1996), 1–27.

[38] Manuel Meier, Paul Streli, Andreas Fender, and Christian Holz. 2021. TapID: Rapid touch interaction in virtual reality using wearable sensing. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 519–528.

[39] 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.

[40] Aunnoy K Mutasim, Anil Ufuk Batmaz, and Wolfgang Stuerzlinger. 2021. Pinch, click, or dwell: Comparing different selection techniques for eye-gaze-based pointing in virtual reality. In *ACM Symposium on Eye Tracking Research and Applications*. 1–7.

[41] Yeji Park, Jiwan Kim, and Ian Oakley. 2024. The Impact of Gaze and Hand Gesture Complexity on Gaze–Pinch Interaction Performances. In *Adjunct of UbiComp '24*. DOI: http://dx.doi.org/10.1145/3675094.3678990

[42] Ken Pfeuffer, Jason Alexander, Ming Ki Chong, and Hans Gellersen. 2014. Gaze-touch: combining gaze with multi-touch for interaction on the same surface. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 509–518.

[43] Ken Pfeuffer, Hans Gellersen, and Mar Gonzalez-Franco. 2024. Design Principles and Challenges for Gaze + Pinch Interaction in XR. *IEEE Computer Graphics and Applications* (2024). DOI: http://dx.doi.org/10.1109/MCG.2024.3382961

[44] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. 2017. Gaze + Pinch interaction in virtual reality. In *Proceedings of the 5th symposium on spatial user interaction*. 99–108.

[45] 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 Symposium on Eye Tracking Research and Applications*. ACM, 1–10. DOI: http://dx.doi.org/10.1145/3204493.3204541

[46] Robert Rosenberg and Mel Slater. 2002. The chording glove: a glove-based text input device. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 29, 2 (2002), 186–191.

[47] Dario D Salvucci and Joseph H Goldberg. 2000. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on Eye tracking research & applications*. 71–78.

[48] Zhaomou Song, John J. Dudley, and Per Ola Kristensson. 2023. HotGestures: Complementing Command Selection and Use with Delimiter-Free Gesture-Based Shortcuts in Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics* 29, 11 (Nov. 2023), 4600–4610. DOI: http://dx.doi.org/10.1109/TVCG.2023.3320257

[49] 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.

[50] Robyn Speer. 2022. rspeer/wordfreq: v3.0. (Sept. 2022). DOI: http://dx.doi.org/10.5281/zenodo.7199437

[51] 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.

[52] Paul Streli, Jiaxi Jiang, Andreas Rene Fender, Manuel Meier, Hugo Romat, and Christian Holz. 2022. TapType: Ten-finger text entry on everyday surfaces via Bayesian inference. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–16.

[53] Uta Wagner, Andreas Asferg Jacobsen, Tiare Feuchtner, Hans Gellersen, and Ken Pfeuffer. 2024. Eye-Hand Movement of Objects in Near Space Extended Reality. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–13.

[54] Tingjie Wan, Yushi Wei, Rongkai Shi, Junxiao Shen, Per Ola Kristensson, Katie Atkinson, and Hai-Ning Liang. 2024. Design and evaluation of controller-based raycasting methods for efficient alphanumeric and special character entry in virtual reality. *IEEE Transactions on Visualization and Computer Graphics* 30, 9 (2024), 6493–6506.

[55] 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.

[56] 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.

[57] Shumin Zhai and Per Ola Kristensson. 2012. The word-gesture keyboard: reimagining keyboard interaction. *Commun. ACM* 55, 9 (Sept. 2012), 91–101. DOI: http://dx.doi.org/10.1145/2330667.2330689

[58] 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*. 595–606.

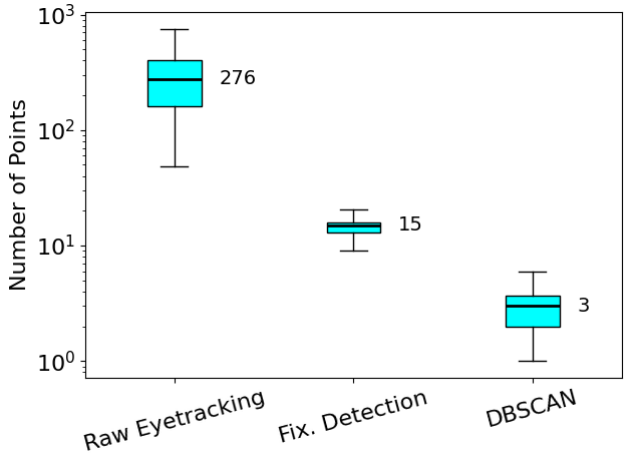


Figure 15: Number of gaze points is reduced significantly through Fixation Detection and then DBSCAN. Data from user study 1; numbers on the side are medians.

A OPTIMIZATION IN SWEYEPINCH DECODING

Optimization: Fixation Detection (I-VT). Our first optimization applies a fixation-detection algorithm to the raw gaze sequence, leveraging the extensive previous research on fixation-detection methods [3, 47]. We use a velocity-thresholding approach (I-VT) because it is efficient and does not require per-user calibration. Pseudocode is presented in Algorithm 2. Conceptually, I-VT labels each gaze point based on its angular velocity relative to the previous valid point. In our experiments, a typical swipe lasts 1.60 ± 0.694 s, corresponding to 320 ± 139 raw gaze samples. After applying I-VT, the number of gaze points (i.e., "fixation" points) decreases to 14.74 ± 2.90.

Optimization: Clustering (DBSCAN). Even after removing saccades, 50+ points can still be expensive for real-time DTW across thousands of candidate templates. Consequently, we further cluster and condense these fixation points via DBSCAN. Let us denote the 3D position (2D spatial coordinates plus 1D timestamp) of a fixation point by $\mathbf{p}_i$. DBSCAN requires two parameters, $\varepsilon_{\text{dbscan}}$ (the neighborhood radius) and *minPts* (the minimum cluster size). Informally, $\mathbf{p}_i$ is a *core point* if it has at least *minPts* neighbors within distance ε. Formally, for each point $\mathbf{p}_i$,

$$\text{neighbors}(\mathbf{p}_i) \;=\; \{\mathbf{p}_j : \|\mathbf{p}_j - \mathbf{p}_i\| \;\le\; \varepsilon_{\text{dbscan}}\}.$$

If $\left|\text{neighbors}(\mathbf{p}_i)\right| \;\ge\; \textit{minPts}$, then $\mathbf{p}_i$ is deemed a core point, and DBSCAN links $\mathbf{p}_i$ to other core points that share neighbors, thus forming a cluster. Any point not belonging to any cluster is labeled as noise. We then replace each cluster of fixation points with its centroid (for example, its mean position), further trimming the size of the gaze sequence. In our tests, DBSCAN reduces the number of points by about an order of magnitude, from 14.74 ± 2.90 down to 2.81 ± 1.28.

Filtering by Starting Letter. Although the above steps significantly reduce the gaze sequence, we still run DTW against thousands of word templates. As an additional heuristic, we *filter candidate words by their initial letter*. Specifically, we take the first centroid $\mathbf{p_1}$ from the DBSCAN output and measure its distance to the center $\ell \in \mathbb{R}^2$ of each letter on the keyboard. If $\|\mathbf{p_1} - \ell\|$ is below a predefined threshold R, we retain only those words whose first letter is ℓ. Formally,

$$\mathcal{V}_{\text{filtered}} \;=\; \Big\{w \in \mathcal{V} \;\Big|\; \big\|\mathbf{p_1} - L(w_1)\big\| \le R\Big\},$$

where $\mathcal{V}$ is the entire vocabulary of candidate words, w_1 is the first letter of word w, and $L(\cdot)$ maps a letter to its canonical keyboard coordinates. This basic geometric filter sharply narrows down $\mathcal{V}_{\text{filtered}}$ before we proceed with DTW.

Cardinality reduction with I-VT and Clustering. The core of *Gaze2Word* is based on DTW, because it is able to align two time series of different lengths and speeds. For two time series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_m)$, DTW runs in $O(nm)$ time. In our case, n corresponds to the number of points in the user's gaze trace, while m corresponds to the number of points in the candidate word's trace (i.e., its length in letters). Because gaze is typically sampled at a high rate (200 Hz in our experiments), n can be large, making *optimizations* that reduce the effective length of the gaze trace crucial. We introduce two main optimizations: *Fixation Detection* and *Clustering*. As shown in Figure 15, these optimizations can reduce the number of gaze points by about two orders of magnitude.

Spatiotemporal DTW: Detailed Derivation. Given a preprocessed gaze trace $\mathbf{g} = \{(x_i, y_i, t_i)\}_{i=1}^{n}$ and a template trace $\mathbf{q} = \{(x'_j, y'_j, t'_j)\}_{j=1}^{m}$, DTW computes the optimal alignment cost via dynamic programming:

$$D(i,j) \;=\; dist\big(\mathbf{g}_i, \mathbf{q}_j\big) + \min\begin{Bmatrix} D(i-1,j), \\ D(i,j-1), \\ D(i-1,j-1) \end{Bmatrix},$$

$$\text{DTW}(\mathbf{X}, \mathbf{Y}) \;=\; D(n,m),$$

with boundary conditions $D(0,0) = 0$, $D(i,0) = \infty$, and $D(0,j) = \infty$ for $i, j > 0$. Here, n and m denote the lengths of sequences $\mathbf{X}$ and $\mathbf{Y}$, respectively.

Adding time and positional index as an additional dimension prevents late samples from aligning to early template points, maintaining temporal monotonicity. Doing so also preserves speed invariance—because t'_j encodes only order, not absolute time. In practice, this reduces pathological alignments that Fréchet- or purely spatial DTW can admit when a single noisy fixation dominates. Our post hoc evaluation shows that including timestamps improved the top candidate match rate from 72.1% to 77.9% (more details in table Table 2). We convert the raw DTW distance $\text{DTW}(\mathbf{g}, \mathbf{t}_w)$ into probabilities through inverting and min-max normalization, as described in Algorithm 1, line 13. Hence, a smaller DTW distance yields a higher value of $p_{\text{dist}}(w; \mathbf{g})$.

B FAT-FINGER AVOIDANCE VIA 2D GAUSSIAN AND CHARACTER-LEVEL N-GRAM

To mitigate these "fat-finger" errors, we combine a 2D Gaussian-based model with a language model to infer the most likely intended

Algorithm 2 Velocity-Thresholding Identification (I-VT)

```
function I-VT(gaze_points)
    for i ← 1 to N do
        if gaze_points[i − 1].valid and gaze_points[i].valid then
            V1 ← gaze_points[i − 1].gazeVector
            V2 ← gaze_points[i].gazeVector
            if AngularVelocity(V1, V2) < 100 deg/s then
                gaze_points[i].type ← "Fixation"
            else
                gaze_points[i].type ← "Saccade"
            end if
        else
            gaze_points[i].type ← "Undefined"
        end if
    end for
    return gaze_points
end function
```

letter. Let $\mathbf{z} = (z_x, z_y)$ be the user's raw input (the tap or the gaze intersection point), and let $\mathbf{c}_\ell = (c_{\ell x}, c_{\ell y})$ be the center of letter ℓ on the virtual keyboard. We define a 2D Gaussian likelihood as:

$$p_{\text{gauss}}(\ell \mid \mathbf{z}) = \frac{1}{2\pi\,\sigma_x\sigma_y} \exp\!\Big(-\tfrac{1}{2}\big(\mathbf{z}-\mathbf{c}_\ell\big)^\top \Sigma^{-1}\big(\mathbf{z}-\mathbf{c}_\ell\big)\Big),$$

where $\Sigma = \text{diag}(\sigma_x^2, \sigma_y^2)$ and (σ_x, σ_y) approximate "key size" or tapping variance. This gives higher probability to letters near $\mathbf{z}$, yet still accommodates nearby keys.

We then blend this likelihood with a character-level n-gram model: $p_{\text{ngram}}(\ell \mid \text{prefix})$, where ℓ is the next character and *prefix* is the typed text thus far (we use a bigram in practice). Let $\alpha \in [0, 1]$ control the relative contribution of the n-gram. The final probability is

$$p(\ell \mid \mathbf{z}, \text{prefix}) = \alpha\, p_{\text{ngram}}(\ell \mid \text{prefix}) + (1-\alpha)\, p_{\text{gauss}}(\ell \mid \mathbf{z}),$$

which yields a higher score for letters favored by both the geometric and language models.

C LATE-TRIGGER DRIFT

Following reports on *Gaze&Pinch* temporal mis-synchrony and late-trigger selection [41, 43], we quantified drift between the intended key center and the actual gaze location at pinch. We found the mean drift magnitude was 21.2 ± 30.6 mm. We mitigate with a *fat-finger* Bayesian letter inference (see Appendix B); 2σ covers $\sim$ 95% of observed drifts. We observe that the drift remained constant across sessions while Gaze&Pinch WPM rose, indicating the model absorbs temporal mis-sync without added user burden.

D AGGREGATED RESULTS FROM PRIOR TEXT-ENTRY STUDIES

This section provides a detailed overview of existing XR text-input techniques evaluated in prior research. Table 5 compiles reported performance metrics across diverse modalities (e.g., controller-based, hand-tracking, gaze-based, other eye swipes), listing average speeds (WPM), error rates, and study details such as participant counts and publication venues. Although this table is by no means complete, we hope it offers the reader a broader context of how recent methods compare with one another and with our proposed techniques.

E PARTICIPANT DEMOGRAPHICS

We provide additional detail on the prior experience of participants in US1, US2, and US3. Figure 16 displays the distribution of self-reported usage frequencies for computer and cellphone keyboards, along with participants' XR exposure and XR keyboard experience. We hope this demographic information helps readers contextualize subsequent performance.

Across US1, US2, and US3 survey responses, participants who reported prior experience with XR systems indicated that their exposure primarily involved gaming, academic projects, software development, public demos, or research, frequently citing devices such as Meta Quest and Apple Vision Pro.

Among those who have used XR keyboards, many described the interaction as unintuitive or inefficient, often attributing challenges to tracking inaccuracies, the absence of haptic feedback, and physical discomfort. However, some participants noted improved usability over time or with the use of more advanced systems such as the Apple Vision Pro.

F ADDITIONAL RESULTS

We present supplementary user study results that were omitted from the primary text. While data presented here did not significantly alter our primary conclusions, they are included here to provide a more complete picture of each user study and to address secondary questions readers may have.

Figure 17 shows the Pareto and NASA TLX results from *US3*—stratifying participants into those who only took part in US3 versus those who also participated in US1.

In our paper, we show WPM and TER from US3 with stratified participants (based on their participation in just US3 or both). Here in the supplementary material, we present Figure 18, showing WPM and TER without stratification. Finally, Figure 19 plots the TER observed over the 30-session period in *User Study 4*. We hope this can provide additional insight into the long-term performance trajectories of individual participants.

Technique	Modality	Speed (WPM)	Error Evaluation Metrics	Error (%)	Number of Participants	Publication	Year of Publication
Raycasting[6]	Controller	~16.65	Total Error Rate (TER)	~11.05%	22	IJVR	2019
Drum-like Keyboard[6]	Controller	~21.01	Total Error Rate (TER)	~12.11%	22	IJVR	2019
Head-directed Input[6]	Controller	~10.83	Total Error Rate (TER)	~10.15%	22	IJVR	2019
Split Keyboard[6]	Controller	~10.17	Total Error Rate (TER)	~8.11%	22	IJVR	2019
Controller + Tap[55]	Controller	14.60	Total Error Rate (TER)	1.65%	12	ISMAR	2019
Touch + Controller[30]	Controller	N/A (faster)	Selection Error Rate	~7.3%	24	TVCG	2023
Raycast + Controller[30]	Controller	11.8% faster	Selection Error Rate	~8.1%	24	TVCG	2023
Standard Qwerty Keyboard (STD)[54]	Controller	~13.23	Total Error Rate (TER)	~8.27%	24	TVCG	2024
Layer-PointSwitch (LPS)[54]	Controller	~12.04	Total Error Rate (TER)	~8.64%	24	TVCG	2024
Layer-ButtonSwitch (LBS)[54]	Controller	~15.94	Total Error Rate (TER)	~4.65%	24	TVCG	2024
Key-ButtonSwitch (KBS)[54]	Controller	~15.08	Total Error Rate (TER)	~4.46%	24	TVCG	2024
Controller + 2D Keys[2]	Controller	14.08	Character Error Rate (CER)	1.91%	28	ICMI	2024
Controller + 3D Keys[2]	Controller	16.52	Character Error Rate (CER)	1.17%	28	ICMI	2024
DesktopKeyboard + NoReposition[13]	Desktop Keyboard	26.3	Character Error Rate (CER)	2.1%	24	IEEE VR	2018
DesktopKeyboard + Reposition[13]	Desktop Keyboard	25.5	Character Error Rate (CER)	2.4%	24	IEEE VR	2018
BlinkType[29]	Gaze + Blink	13.47	Total Error Rate (TER)	10.44%	36	arXiv	2020
Click[40]	Gaze + Controller	N/A	Error Rate	~11%	12	ETRA	2021
Pinch[40]	Gaze + Hand	N/A	Error Rate	~13%	12	ETRA	2021
Gaze + Keyboard Button[12]	Gaze	10.98	Relative Error Rate	8.0%	15	AVI	2012
EyeSwipe (Dwell-free)[24]	Gaze	11.7	Minimum String Distance (MSD) Error	1.31%	10	CHI	2016
Dwell-time Gaze Typing[24]	Gaze	9.5	Minimum String Distance (MSD) Error	1.01%	10	CHI	2016
Static Dwell Gaze Typing[39]	Gaze	10.62	Total Error Rate (TER)	16.83%	17	CHI	2017
Cascading Dwell Gaze Typing[39]	Gaze	12.39	Total Error Rate (TER)	11.08%	17	CHI	2017
Flat Keyboard + Sitting + Dwell[45]	Gaze	9.36	Rate of Backspace Activation (RBA)	2.0%	16	ETRA	2018
Flat Keyboard + Sitting + Click[45]	Gaze	10.15	RBA	7.0%	16	ETRA	2018
Flat Keyboard + Biking + Dwell[45]	Gaze	8.07	RBA	4.0%	16	ETRA	2018
Flat Keyboard + Biking + Click[45]	Gaze	8.58	RBA	8.0%	16	ETRA	2018
Curved Keyboard + Sitting + Dwell[45]	Gaze	7.48	RBA	6.0%	16	ETRA	2018
Curved Keyboard + Sitting + Click[45]	Gaze	9.15	RBA	3.0%	16	ETRA	2018
Curved Keyboard + Biking + Dwell[45]	Gaze	6.77	RBA	4.0%	16	ETRA	2018
Curved Keyboard + Biking + Click[45]	Gaze	8.29	RBA	3.0%	16	ETRA	2018
Dwell[40]	Gaze	N/A	Error Rate	~7%	12	ETRA	2021
Eye-typing with Dwell (OptiKey)[4]	Gaze	13.0	Uncorrected Error Rate	5.7%	18	PLOS ONE	2021
Dwell-Typing[31]	Gaze	9.50	Total Error Rate (TER)	4.06%	16	ETRA	2022
GlanceWriter (Exp I)[9]	Gaze	10.89	Word Error Rate (WER)	2.71%	14	CHI	2023
EyeSwipe[9]	Gaze	6.49	Word Error Rate (WER)	6.85%	14	CHI	2023
GlanceWriter (Exp II)[9]	Gaze	9.54	Word Error Rate (WER)	12.89%	12	CHI	2023
Tobii Communicator 5[9]	Gaze	7.41	Word Error Rate (WER)	16.32%	12	CHI	2023
S-Gaze&Finger[31]	Hand + Gaze	10.66	Total Error Rate (TER)	2.9%	16	ETRA	2022
S-Gaze&Hand[31]	Hand + Gaze	9.49	Total Error Rate (TER)	2.98%	16	ETRA	2022
Speedup (Gaze-Assisted)[58]	Hand + Gaze	17.6	Word Error Rate (WER)	1.01%	12	IUI	2023
Gaussian Speedup (Gaze-Assisted)[58]	Hand + Gaze	17.1	Word Error Rate (WER)	0.71%	12	IUI	2023
Hand + Tap[55]	Hand	6.74	Total Error Rate (TER)	6.48%	12	ISMAR	2019
AirTap[31]	Hand	11.37	Total Error Rate (TER)	3.54%	16	ETRA	2022
Touch + Hand[30]	Hand	N/A	Selection Error Rate	~7.3%	24	TVCG	2023
Raycast + Hand[30]	Hand	slower	Selection Error Rate	~8.1%	24	TVCG	2023
Wrist-Only Gesture Typing[58]	Hand	16.4	Word Error Rate (WER)	0.98%	12	IUI	2023
Midair Keyboard Only (NoSpeech)[1]	Hand-tracking	11.1	Character Error Rate (CER)	1.2%	18	TVCG	2020
Hand-tracking + 2D Keys[2]	Hand-tracking	12.78	Character Error Rate (CER)	0.98%	28	ICMI	2024
Hand-tracking + 3D Keys[2]	Hand-tracking	14.73	Character Error Rate (CER)	0.85%	28	ICMI	2024
HGaze Typing (Head + Gaze)[11]	Head + Gaze	11.22	Minimum String Distance (MSD) Error	~0.37%	10	ETRA	2021
Hybrid + Tap[55]	Head + Hand	8.53	Total Error Rate (TER)	3.85%	12	ISMAR	2019
Head + Keyboard Button[12]	Head	4.42	Relative Error Rate	3.8%	15	AVI	2012
Head + Mouth Open Gesture[12]	Head	3.07	Relative Error Rate	6.0%	13	AVI	2012
Head + Brows Up Gesture[12]	Head	2.85	Relative Error Rate	21.0%	13	AVI	2012
TapType[56]	Head	15.58	Total Error Rate (TER)	2.02%	6	CHI	2017
DwellType[56]	Head	10.59	Total Error Rate (TER)	3.69%	6	CHI	2017
GestureType[56]	Head	19.04	Total Error Rate (TER)	4.21%	6	CHI	2017
GestureType (trained)[56]	Head	24.73	Total Error Rate (TER)	5.82%	12	CHI	2017
Head + Tap[55]	Head	5.62	Total Error Rate (TER)	1.06%	12	ISMAR	2019
DwellType[29]	Head	11.65	Total Error Rate (TER)	9.64%	36	arXiv	2020
NeckType[29]	Head	11.18	Total Error Rate (TER)	9.04%	36	arXiv	2020
Speech + Midair Keyboard[1]	Speech + Hand-tracking	27.9	Character Error Rate (CER)	0.5%	18	TVCG	2020
TouchscreenKeyboard + NoReposition[13]	Touchscreen Keyboard	11.6	Character Error Rate (CER)	2.7%	24	IEEE VR	2018
TouchscreenKeyboard + Reposition[13]	Touchscreen Keyboard	8.8	Character Error Rate (CER)	3.6%	24	IEEE VR	2018

Table 5: XR text-input techniques proposed over the years, sorted first by Modality and then by Year of Publication.

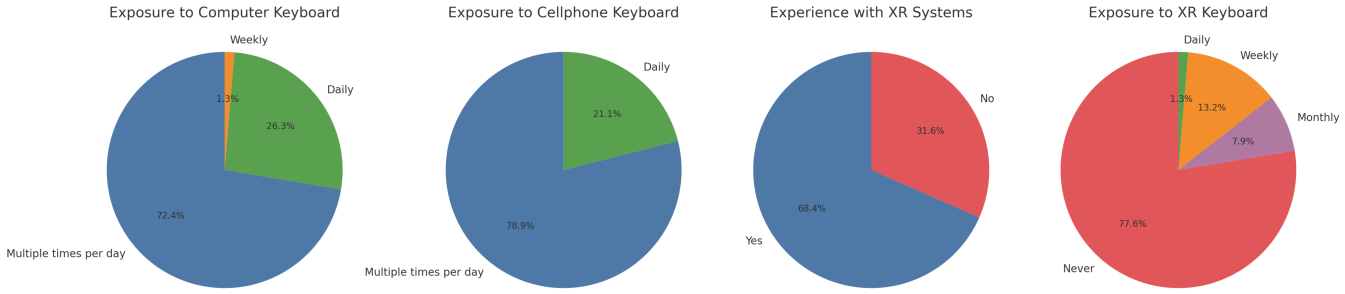


Figure 16: Participant demographics through US1, US2 and US3, showing participants' reported exposure to computer keyboards, cellphone keyboards, participants' experience with XR systems and exposure to XR keyboard.

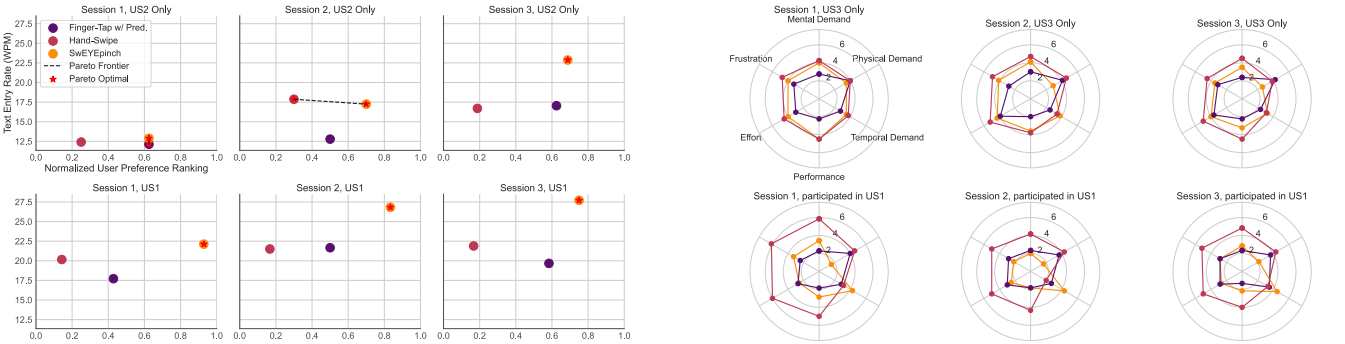


Figure 17: Left: Pareto Frontier. Right: NASA TLX. Data are shown across all *three* sessions from *US3*. "US3 Only" includes participants who only participated in US3. Bottom row "US1" includes participants who have also participated in US1.

User	Learning rate (WPM/session)				Median WPM		
	1–10	11–20	21–30	All	First	Best	ΔWPM
P401	1.93	1.14	0.263	1.02	26.7	60.6	33.9
P402	1.33	1.77	1.11	1.29	19.8	64.7	44.9
P403	1.92	1.67	0.524	0.943	8.35	45.4	35.8
P404	0.95	1.82	0.961	0.807	22.0	46.2	24.1
P405	1.53	1.12	0.774	1.14	9.8	56.5	46.8
P406	2.43	1.08	1.43	1.65	5.5	45.9	40.4
P407	1.51	0.38	2.98	1.62	8.0	57.0	49.1
P408	2.35	1.45	1.65	1.09	6.4	48.9	42.5
P409	1.16	1.02	0.118	0.767	15.1	60.9	45.8

Table 6: Per-user learning rates (WPM/session) in 10-session blocks and overall, and median WPM at first vs. best session with gains.

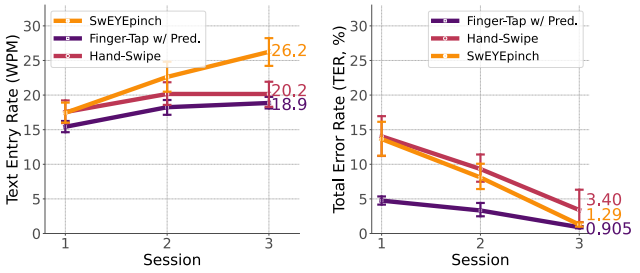


Figure 18: *left*: Average text entry speeds (WPM). *right*: TER. Data are shown across all *three* sessions from *US3*.

F.1 Summary of Qualitative User Feedback

In this section, we present the excerpts from participant comments and a thematic breakdown of user impressions across different typing methods, drawn from both *US1* and *US3*. Table 7 summarizes key feedback themes (e.g., ease of use, speed, error handling) for Finger-Tap, Gaze&Pinch, and SwEYEpinch-Basic in *US1*. Representative user quotes appear in Table 8.

For *US3*, Table 9 and Table 10 provides a similar thematic summary contrasting Finger-Tap with Prediction&Completion, Hand-Swipe, and SwEYEpinch. Table 10 contains additional direct quotes illustrating how participants perceived each method's advantages and drawbacks. We hope these summaries and excerpts can help contextualize our quantitative findings.

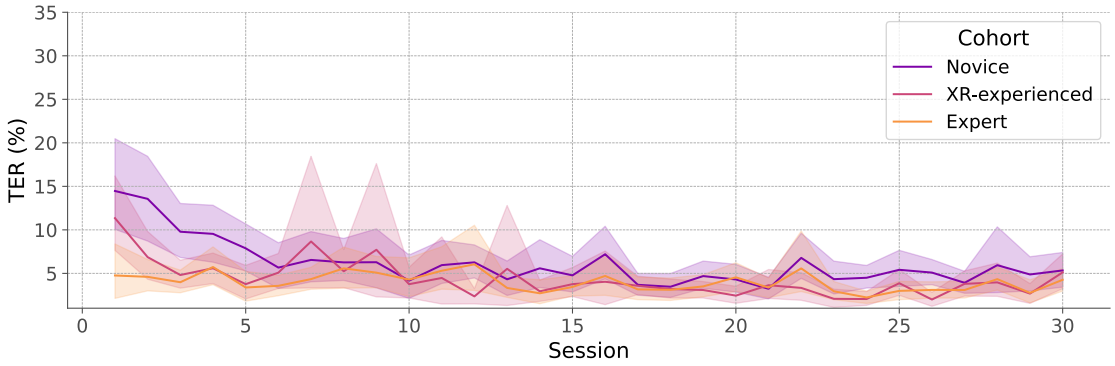


Figure 19: TER for the three user cohorts in *US4*, across the 30 sessions.

Theme	Finger-Tap	Gaze&Pinch	SwEYEpinch-Basic
Ease of Use / Intuitiveness	Familiar, intuitive like mobile typing; easy to correct errors	Requires learning curve; hard to control gaze and pinch coordination	Easy after practice; auto-suggestion helps users improve speed
Physical Effort	Most physically demanding; arm fatigue common due to hand position	Less physically demanding than Finger-Tap, but tiring over long use	Least physically demanding; only eye and light hand gestures needed
Mental Demand / Fatigue	Low cognitive load; users could monitor text while typing	High mental load; gaze focus and hand coordination stressful	Moderate mental demand; stressful for long or complex words
Accuracy / Reliability	High accuracy; hand tracking occasionally caused unintended double taps	Least accurate; heavily dependent on calibration and tracking quality	Moderate accuracy; autocorrect can help but also cause unintended words
Calibration Sensitivity	Less sensitive; works even with some hand tracking drift	Highly sensitive; even minor tracking drift ruins input	Somewhat sensitive; tracking drift affects swipe path recognition
Error Recovery	Easy to fix errors manually	Error correction slow and frustrating; often required full retyping	Autocorrect helpful, but hard to fix partial mistakes
Speed / Efficiency	Slowest due to physical input speed	Slow due to gaze/pinch coordination	Fastest once mastered; boosted by predictive text
Learning Curve	Very low; resembles real-life typing	Steep; most users struggled to master it in short time	Moderate; improved notably with repeated use
User Preference Trend	Most reliable but tiring; good fallback method	Least preferred due to frustration and tracking issues	Most preferred overall for speed and comfort

Table 7: Summary of user feedback for Finger-Tap, Gaze&Pinch, and SwEYEpinch-Basic (US1).

Typing Method	Theme	Representative Quote
Finger-Tap	Physical Effort	"Finger-Tap was easy but physically demanding."
		"Finger tap required a lot of upper arm strain in order for the computer to pick up my hand movements."
	Accuracy / Control	"Finger tap is the easiest because it resembles the keyboard most."
		"Easy to fix the mistake, unlike the other two methods."
	Familiarity	"Finger-Tap is still the most intuitive way to type... muscle memory typing on a keyboard."
Gaze&Pinch	Calibration Issues	"Gaze&pinch was hardest due to eye tracking not being accurate."
		"If the calibration was even slightly off, it made the task extremely frustrating."
	Physical / Mental Demand	"Gaze&Pinch was the most mentally and physically exhausting."
	Improvement with Familiarity	"I would rank Gaze&Pinch over Finger-Tap because I am familiar with the distance to trigger the pinch action and thus typing speed is improved."
SwEYEpinch-Basic	Speed / Ease	"Swipe can be the fastest and easiest because you don't have to focus on the letters."
		"Once I mastered the coordination, swipe was the simplest to use."
	Autocorrect / Error Handling	"Swipe was fun and quite easy to fix when it went wrong."
		"Swipe had a hard time recognizing some common words like 'the' and 'strawberries'."
	Learning Curve / Fatigue	"Swipe was initially hard to get used to, but it became a lot less demanding once I understood how it works."
		"Swipe required a lot of effort because often if I would misread the word... it required effort to retype it."

Table 8: Representative user quotes for US1

Theme	Finger-Tap w/ Pred.	Hand-Swipe	SwEYEpinch
Ease of Use / Intuitiveness	Familiar and similar to phone typing; low learning curve	Not intuitive for many; difficult gesture control	Intuitive after learning; feels futuristic for some
Physical Effort	Physically taxing; arm fatigue common	Most physically exhausting; holding hand up is tiring	Least physically demanding; only requires eye movement
Mental Demand / Fatigue	Low mental effort; simple concept	High mental demand from coordinating swipes and gestures	Medium; some strain from focusing gaze
Accuracy / Reliability	Reliable; occasional finger tracking issues	Inconsistent detection; trouble with long/complex words	Good when calibrated; errors at periphery of vision
Calibration Sensitivity	Less sensitive to misalignment	Sensitive to hand angle and distance	Requires precise calibration for gaze to work well
Error Recovery	Easy to correct manually	Difficult to delete or redo words; no easy back gesture	Better with autocomplete, but hard to recover from mistakes
Speed / Efficiency	Slower but reliable	Slower due to gesture complexity	Fastest when functioning correctly; improved by predictive text
Learning Curve	Very low; familiar typing behavior	Steep; requires practice and good calibration	Moderate; intuitive after learning curve
User Preference Trend	Preferred for familiarity and control	Least preferred due to frustration and physical effort	Frequently preferred for long-term use if calibrated

Table 9: Summary of user feedback for *Finger-Tap*, *Hand-Swipe*, and *SwEYEpinch* (US3).

Typing Method	Theme	Representative Quote
Finger-Tap w/ Pred.	Physical Effort	"Finger tap hurts my arms." "Finger tap required a lot of upper arm strain..."
	Familiarity	"Finger tap was most familiar and intuitive." "Finger tap mimics real-life typing and is easiest to use."
	Accuracy / Reliability	"Most reliable, but sometimes missed my finger."
Hand-Swipe	Physical Demand	"Hand-Swipe is physically exhausting; my arms got tired." "Holding hand up and pinching for each word is tedious."
	Accuracy / Control	"Not intuitive at all." "High learning curve and frequent gesture misfires."
	Frustration	"Hand-Swipe was the most frustrating method."
SwEYEpinch	Speed / Comfort	"Felt like the future; quick and minimal effort." "Eye swipe was quite intuitive after learning."
	Calibration Issues	"Only works well with proper calibration." "Had issues tracking eyes at edge of keyboard."
	Prediction Benefit	"Autocomplete made eye swipe very fast." "Suggestions helped speed up typing significantly."

Table 10: Representative User Quotes from US3

G QUESTIONNAIRE

We use the following survey for our study. The exact text of each question as presented to the participants is below. The content here is for US1. The US2 and US3 surveys are identical except for the text entry techniques covered.

(1) Age
(2) Gender
(3) Right-handed or left-handed?
(4) I have normal or corrected-to-normal vision (with or without glasses).
(5) Strabismus (crossed-eyes)?
(6) I use a computer keyboard...
(7) I use a cellphone keyboard...
(8) I have experience using Augmented Reality/Virtual Reality systems
(9) If you answered "Yes" to the previous question, please explain your experience.
(10) I use Augmented Reality/Virtual Reality keyboards...
(11) If you have used an Augmented Reality/Virtual Reality keyboard, please tell us more about your experience (e.g., how intuitive/efficient do you find them to be)?
(12) Mental Demand: how mentally demanding was the task?: Finger-Tap
(13) Mental Demand: how mentally demanding was the task?: Gaze&Pinch
(14) Mental Demand: how mentally demanding was the task?: Swipe
(15) Physical Demand: how physically demanding was the task?: Finger-Tap
(16) Physical Demand: how physically demanding was the task?: Gaze&Pinch
(17) Physical Demand: how physically demanding was the task?: Swipe
(18) Temporal Demand: how hurried or rushed was the pace of the task?: Finger-Tap
(19) Temporal Demand: how hurried or rushed was the pace of the task?: Gaze&Pinch
(20) Temporal Demand: how hurried or rushed was the pace of the task?: Swipe
(21) Performance: how successful were you in accomplishing what you were asked to do?: Finger-Tap
(22) Performance: how successful were you in accomplishing what you were asked to do?: Gaze&Pinch
(23) Performance: how successful were you in accomplishing what you were asked to do?: Swipe
(24) Effort: how hard did you have to work to accomplish your level of performance?: Finger-Tap
(25) Effort: how hard did you have to work to accomplish your level of performance?: Gaze&Pinch
(26) Effort: how hard did you have to work to accomplish your level of performance?: Swipe
(27) Frustration: how insecure, discouraged, irritated, stressed, and annoyed were you?: Finger-Tap
(28) Frustration: how insecure, discouraged, irritated, stressed, and annoyed were you?: Gaze&Pinch
(29) Frustration: how insecure, discouraged, irritated, stressed, and annoyed were you?: Swipe
(30) Please rank the styles of typing by your overall experience with them.: Finger-Tap
(31) Please rank the styles of typing by your overall experience with them.: Gaze&Pinch
(32) Please rank the styles of typing by your overall experience with them.: Swipe
(33) Please explain your rankings.
(34) Do you have any additional comment about the three typing methods (Finger-Tap, Gaze&Pinch, Swipe)?