EyeO: Autocalibrating Gaze Output with Gaze Input for Gaze Typing

Akanksha Saran*
Sony AI
San Francisco, CA, USA
akanksha.saran@sony.com

Jacob Alber
Microsoft Research
New York, NY, USA
jacob.alber@microsoft.com

Cyril Zhang
Microsoft Research
New York, NY, USA
cyrilzhang@microsoft.com

Ann Paradiso
Microsoft Research
Redmond, WA, USA
annpar@microsoft.com

Danielle Bragg
Microsoft Research
Cambridge, MA, USA
danielle.bragg@microsoft.com

John Langford
Microsoft Research
New York, NY, USA
jcl@microsoft.com

Abstract

Gaze tracking devices have the potential to expand interactivity greatly, yet miscalibration remains a significant barrier to use. As devices miscalibrate, people tend to compensate by intentionally offsetting their gaze, which makes detecting miscalibration from eye signals difficult. To help address this problem, we propose a novel approach to seamless calibration based on the insight that the system's model of eye gaze can be updated during reading (user does not compensate) to improve calibration for typing (user might compensate). To explore this approach, we built an autocalibrating gaze typing prototype called EyeO and ran a user study with 20 participants. Our user study results suggest that seamless autocalibration can significantly improve typing efficiency and user experience.

CCS Concepts

• **Human-centered computing** → **Interaction techniques**; **Interaction devices**; **Keyboards**; **Displays and imagers**.

Keywords

gaze tracking, eye typing, gaze typing, autocalibration, miscalibration, implicit calibration

ACM Reference Format:
Akanksha Saran, Jacob Alber, Cyril Zhang, Ann Paradiso, Danielle Bragg, and John Langford. 2025. EyeO: Autocalibrating Gaze Output with Gaze Input for Gaze Typing. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25), April 26–May 01, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3706599.3720090

1 Introduction

Eye tracking technology has emerged as a valuable tool in accessibility, augmented reality (AR), virtual reality (VR), robotics, and gaming [28, 41]. A key use case is gaze typing, enabling text input via gaze on an on-screen keyboard—a hands-free method of communication widely adopted in both general and assistive contexts. Beyond current applications, eye tracking holds the potential to enhance interaction by understanding user attention and personalizing interfaces [9].

Despite this potential, calibration difficulties remain a major barrier to use [1, 2, 13, 19, 20, 40]. Calibration is a process that establishes a relationship between the user's gaze and the corresponding screen coordinates. It is traditionally achieved through an explicit guided process of looking at visual targets, separate from the downstream application the user might be controlling with their gaze. While effective, this explicit calibration approach is tedious, time-consuming, and sensitive to variations in lighting, head position, and other factors, often necessitating frequent recalibrations [16, 42, 43]. This challenge is particularly acute for users with motor impairments who may struggle with traditional calibration protocols, and have to recalibrate their eye tracker between 3-10 times per day [9]. Miscalibration not only degrades system performance but also burdens users, who must compensate for inaccuracies, leading to additional cognitive load.

Although automatic miscalibration detection and correction could address this issue, users' compensatory behaviors—such as intentionally offsetting their gaze—complicate such efforts. For instance, in dwell-based gaze typing [24, 29], users may deliberately glance at adjacent keys to activate the intended key, further confounding miscalibration detection. These behaviors, while enabling task completion, increase user effort and reduce system usability.

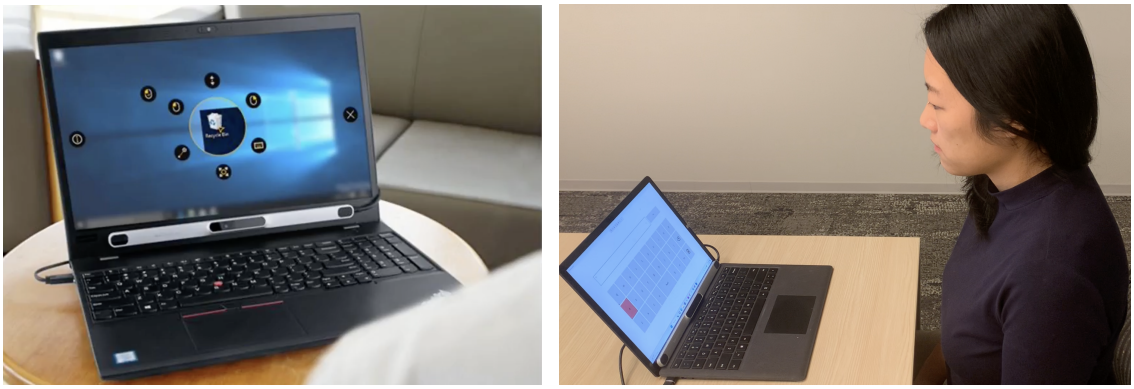
In this work, we introduce EyeO, a novel autocalibrating gaze typing prototype that addresses these challenges. EyeO leverages a key insight: while users compensate for miscalibration during typing, they do not compensate when reading previously typed text. By detecting gaze behavior during reading, EyeO dynamically updates calibration to reduce the need for manual recalibration. The prototype tracks fixations on the last typed character displayed on the screen, comparing the predicted gaze with the character's screen location to estimate and correct calibration offsets. EyeO aims to improve the gaze typing experience by seamlessly adjusting calibration in real-time, thereby reducing the need for manual recalibration and offering a more natural and efficient interaction.

To evaluate EyeO, we conducted a user study with 20 participants examining its impact on typing efficiency, error reduction, and user

*Work done while at Microsoft Research New York.

CHI EA '25, Yokohama, Japan

ACM ISBN 979-8-4007-1395-8/25/04
https://doi.org/10.1145/3706599.3720090



(a) Windows Eye Control application (b) A user operating a customized gaze typing application

Figure 1: Windows applications for (a) visual PC control and (b) gaze typing — both leveraging user's eye movements tracked via an external Tobii eye tracking device.

experience. Our findings demonstrate significant improvements in typing speed and satisfaction compared to the static, explicit calibration approach. EyeO's seamless calibration holds promise for a wide range of users, from those with motor impairments relying on gaze typing for communication to gamers and AR/VR users seeking enhanced interaction.

Our contributions are:

- A novel gaze typing autocalibration algorithm leveraging behavioral differences between typing and reading to correct miscalibration.
- A practical implementation of this algorithm in the EyeO prototype, with the potential to adapt to existing keyboard layouts and eye trackers.
- A user study validating EyeO's effectiveness, highlighting its potential to improve performance and user experience for gaze-based systems.

2 Background and Related Work

We provide a brief overview of prior approaches for explicit and implicit calibration of eye trackers (Sec. 2.1) which rely either on additional external hardware or specialized display designs. Our work overcomes the shortcomings and design restrictions of these prior methods to offer a novel approach for seamless autocalibration by learning from differences in eye gaze between typing and reading. We also briefly review prior findings on the reading behavior of users during gaze typing in Sec. 2.2, which we leverage in our autocalibration algorithm.

2.1 Calibration Methods for Eye Trackers

Calibration is essential in eye tracking systems, mapping gaze to screen coordinates. Traditional methods involve *explicit* calibration, requiring users to fixate on targets like a 9-point grid, which can be time-intensive, interrupt the flow of the task at hand, and are quite uncomfortable for some users, particularly those with motor impairments [9, 18, 20]. To improve user experience, simplified *explicit* calibration techniques have been proposed, such as using fewer [15] or natural targets [37, 38], but these still depend on user cooperation, changes to the keyboard design, and sometimes custom hardware.

Alternatively, *implicit* calibration techniques eliminate the need for explicit user input. These methods leverage saliency maps to infer gaze fixations [12, 13, 20, 40] or use correlations with dynamic stimuli like smooth pursuits [1, 2, 30] or mouse clicks [10, 19]. For example, calibration-free systems have used moving text [21] or clustered, outward-moving characters [22] for gaze typing. However, these approaches often depend on specific interfaces or additional apparatus to take user clicks as input, limiting their flexibility for diverse hands-free applications.

Unlike these prior methods, our approach enables real-time autocalibration during natural gaze interactions without explicit user input or specialized setups, making it adaptable to existing keyboard layouts and eye tracking devices.

2.2 Reading Behavior during Gaze Typing

Research on gaze scan paths shows that, when reading textbooks or digital text for comprehension, people fixate near the center of words [31, 39]. However, gaze typing includes an additional factor: the user is not only reading but also producing the text letter by letter. During text production, the user's immediate concern may be accuracy at a micro-level (the last character typed) rather than fluent reading of entire words or sentences.

Multiple studies and review articles on gaze-based text entry have documented that users frequently verify the result of their input after selecting each character [17, 25–27]. This verification behavior implicitly suggests that users often fixate on the last typed character or the immediate vicinity of it as they proceed. Majaranta and Räihä [25] discuss the importance of feedback mechanisms and usability concerns in eye typing systems. They highlight that users rely on immediate visual feedback to confirm correct input and make corrections, which leads to users often momentarily fixating on what they have just entered. Majaranta and Räihä [26] further elaborate that during dwell-based typing, users rely on looking back

at the typed text to ensure accuracy. Our work leverages this user pattern to detect the offset in predicted gaze for autocalibration.

3 EyeO Prototype

Eye gaze systems typically require users to stop gaze typing tasks to explicitly recalibrate a miscalibrated eye tracker. To implicitly address such task interruption without any change to the keyboard layout, we present EyeO, which dynamically recalibrates the eye tracker during typing without interrupting the user. By leveraging differences in gaze behavior during reading (perception) versus typing (control), EyeO uses the user's natural fixations on the last typed character to update calibration offsets in real-time. This approach eliminates the need for manual recalibration and ensures smooth interaction.

3.1 System Design

EyeO builds on the existing visual keyboard part of the widely available Windows Eye Control application [4] (Fig. 2) and adds dynamic calibration updates in real-time. The system uses a Tobii PCEye infrared tracker (60 Hz sampling rate) connected to a Windows 11 laptop, enabling gaze-controlled typing. The tracker can be calibrated via a standard 9-point calibration software available with the purchase of the tracker. A user's gaze location as detected by the eye tracker is displayed as a red dot on the screen ((Fig. 2). If they fixate on a key for 50 ms, the system initiates a dwell timer of 400ms (Fig. 2(a),(b)). When the timer finishes, the user receives visual feedback that the character on the key has been typed: the key turns red (Fig. 2(c)) and the letter is added to the text box at the top of the screen (Fig. 2(d)).

In this work, we use the visual keyboard layout on a 14 inch Lenovo laptop screen. The keyboard implementation we used is available via the open-source Windows community toolkit [6] and our autocalibration algorithm is added to the keyboard functionality via a UWP application available as part of Microsoft's open-source Gaze Interaction Library [5]. The software for our autocalibration algorithm was developed in Python, which received the 2D gaze coordinates in real-time from the UWP application via Google's remote procedure call (RPC) protocol. This allowed Python scripts to access gaze data, analyze it, and apply the necessary miscalibration corrections to display back on the visual keyboard application in real-time (~ 60 Hz).

3.2 Autocalibration Algorithm

The autocalibration algorithm implemented as part of the EyeO prototype dynamically corrects calibration by detecting gaze offsets during reading. When a user fixates on the last typed character, the system compares the predicted gaze position to the center location of the character typed on the screen. Discrepancies are calculated as calibration offsets, which are smoothed over time using a moving average. These offsets are continuously applied to adjust gaze predictions in real-time, ensuring seamless calibration during ongoing reading and typing tasks. This approach eliminates the need for manual recalibration and adapts to changes in user behavior or environmental conditions. Figure 3 provides a simplified schematic of the procedure, while Algorithm 1 presents it in full detail. Additional details about the hyperparameters used in the Algorithm are described in Appendix A.1.3.

Algorithm 1 EyeO Autocalibration for Gaze Typing

Require: Stream of gaze coordinates (x_t, y_t) detected by an eye tracker; window size w for running average; calibration error bound b; calibration zone threshold τ; y-coordinate for lower boundary of text box $y_{\text{text\_box\_bottom}}$

Initialize $n_{\text{char}} := 0$ // number of characters visible in text box
Initialize $\epsilon_{x_0} := 0, \epsilon_{y_0} := 0$ // calibration error in x and y directions
for raw gaze coordinates (x_t, y_t): **do**
  Receive $n_{\text{char}}$from typing application
  if $y < y_{\text{text\_box\_bottom}}$ **and** $n_{\text{char}} > 0$ **then**
    Receive location of last type character (x_c, y_c) from typing application
    if fixation detected **and** $\sqrt{(x_c - x_t)^2 + (y_c - y_t)^2} < \tau$ **then**
      $\delta_{x_t} := x_c - x_t, \quad \delta_{y_t} := y_c - y_t$ // infer user is reading at (x_c, y_c)
      $\bar{\delta}_{x_t} := \frac{1}{\min(w,t)} \sum_{j=\max(0,t-w+1)}^{t} \delta_{x_t}, \quad \bar{\delta}_{y_t} := \frac{1}{\min(w,t)} \sum_{j=\max(0,t-w+1)}^{t} \delta_{y_t}$ // sliding window estimate
      $\epsilon_{x_t} := \text{clip}(\bar{\delta}_{x_t}, -b, b), \quad \epsilon_{y_t} := \text{clip}(\bar{\delta}_{y_t}, -b, b)$ // clip to maximum allowed offset
    end if
  else
    $\epsilon_{x_t} := \epsilon_{x_{t-1}}, \quad \epsilon_{y_t} = \epsilon_{y_{t-1}}$ // infer user is typing; keep current error calibration
  end if
  return calibrated gaze coordinates $(\widehat{x}_t, \widehat{y}_t) := (x_t + \epsilon_{x_t} + y_t + \epsilon_{y_t})$
end for

4 User Study

To explore the effectiveness of our autocalibration technique and how it shapes a user's gaze typing experience, we ran an in-lab user study with IRB approval. We compared EyeO to a standard manual calibration control (which does not autocorrect in the case of miscalibration). After standard calibration, miscalibration of different amounts was purposefully introduced to evaluate the typing experience of users under such conditions.

4.1 Participants

We recruited 20 participants (14 male, 6 female, aged 18–55) with normal (20/20 vision) or mild to moderate corrected-to-normal vision (20/40 to 20/160). Two participants wore glasses, and two had prior experience with gaze typing. None of the participants had color blindness or any other eye conditions. All of them had at least a university graduate-level education, ensuring a high level of English reading literacy as a proxy for their reading and interpretation skills. Each session lasted approximately 60 minutes with the flexibility to take breaks as needed. Participants were compensated for their time. One participant found the typing experience very uncomfortable

Figure 2: Screenshots of the system's visual keyboard display. A user can activate a key (i.e. type a character) on the visual keyboard by dwelling on it for a fixed duration of time (400ms). Here we show the snippets of visual feedback (in order) that a user receives when successfully typing a character with their eye gaze: ((a),(b)) launching a timer during the dwelling period which is displayed as a shrinking green rectangle, (c) activating the key for the letter 'h' if their eyes are still gazing at that key by the the end of the timer (depicted as the key turning red), (d) the character on the activated key is displayed at the top of the screen.

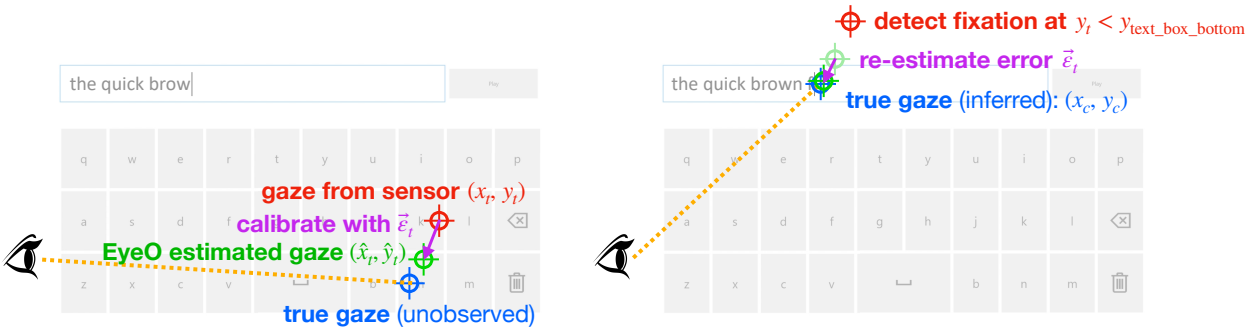


Figure 3: Simplified diagram of Algorithm 1, depicting the input and output modes of the EyeO auto-calibrating interface. *Left:* While the user is typing (lines 12-14), the algorithm receives a stream of miscalibrated sensor inputs (x_t, y_t), and sends a calibrated sequence $(\widehat{x}_t, \widehat{y}_t) := (x_t + \epsilon_{x_t}, y_t + \epsilon_{y_t})$ to the interface, adjusted using the current error estimates. *Right:* While the user is consuming system output (lines 5-11), the system infers that the user is looking at (x_c, y_c), the location of the last typed character, and uses this to *update* the error estimates $(\epsilon_{x_t}, \epsilon_{y_t})$ used for calibration. This inference only occurs when fixation is detected within a threshold τ.

during the practice session and opted out of the study. We thus report results for 19 participants.

4.2 System Setup

Participants were seated comfortably at a distance of approximately 75cm from the laptop screen and head movements were not restrained to encourage natural gaze interaction. Two typing systems were evaluated: (1) EyeO with dynamic autocalibration and (2) a control that relied on traditional calibration alone. Both systems used the same visual keyboard layout and gaze typing application. All manual calibrations were performed via the standard Tobii SDK 9-point calibration procedure. All tests were conducted in a well-lit white-light room.

4.3 Procedure

The experiment consisted of an initial calibration procedure, a practice round, and trials of both EyeO and the control (within-subjects study). The order of system trials was counterbalanced across participants. During each trial, participants typed five random phrases from a standard corpus (MacKenzie and Soukoreff phrase corpus [23]) under systematically induced miscalibration scenarios: (1) 0 (no miscalibration), (2) +75 pixels in the x direction, (3) -75 pixels in the x direction, (4) +75 pixels in the y direction, (3) -75 pixels in the y direction. These five miscalibration amounts were counterbalanced across different users, i.e. all users typed with the same five miscalibration levels but in varying orders. The order of presented miscalibration amounts was balanced across participants and system conditions using the Latin Square technique.

To better understand if EyeO led to a more positive user experience over the control system, we asked participants to rate each system in terms of the mental workload required (at the end of each of the two system trials) using the NASA TLX questionnaire [11] (survey questions are shared as part of the supplementary materials). At the end of the study, we also asked them to share feedback in written form about their gaze typing experience between the two systems, and if they preferred one over the other (the feedback form is shared as part of the supplementary materials). To further assess the effectiveness of EyeO autocalibration for gaze typing, we computed typing efficiency in terms of typing speed (characters per minute), number of backspaces, and abort frequency (number of sessions where the user gave up phrase typing) as quantitative performance metrics.

4.4 Results

4.4.1 Typing Efficiency. We found that EyeO exhibited faster typing speeds (characters/minute) (Fig. 4(a); $p < 0.05$), lower use of backspaces (Fig. 4(b); $p = 0.53$), and lower abort frequency (Fig. 4(c); $p < 0.05$) in comparison to the static control system. Differences between typing speed and abort frequency were statistically significant (Fig. 4(a),(c)). Considering gaze data for EyeO after the first reading attempt (when the system gets the first opportunity to autocalibrate), differences for typing speed and number of backspaces used were even larger (Fig. 4(d),(e)). With this consideration, typing speed is strikingly different between the two systems ($p < 0.001$), hinting towards the effectiveness of EyeO's autocalibration technique.

In Table. 1, we show that EyeO on average improves over the initial 9-point calibration. However, the results are not statistically significant (typing speed: $p = 0.97$, backspace count: $p = 0.53$).

Table 1: Mean and standard error of typing efficiency metrics for the 0-miscalibration case.

System	Typing Speed (chars/min)	Backspace Count
Control	23.72 ± 1.38	1.22 ± 0.49
EyeO	23.81 ± 1.70	0.75 ± 0.46

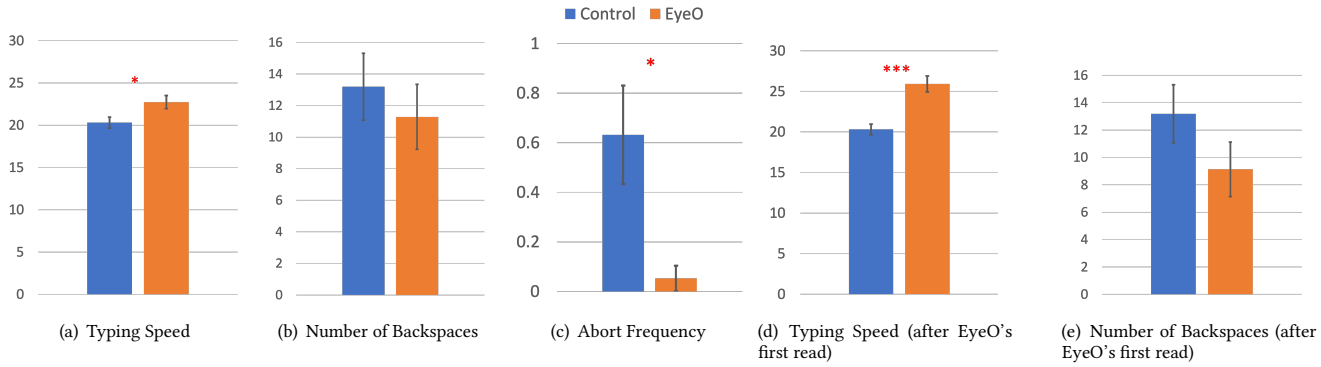
This implies that cases with large miscalibration contribute to more prominent gains for EyeO. We use data from the 0-miscalibration case for both systems to compute these values. No sessions were aborted for any of the two systems in this condition.

4.4.2 Mental Workload. Participants consistently rated EyeO more favorably than the static control (manual calibration) in the NASA TLX survey. This survey assesses a system's mental workload in terms of mental demand, physical demand, temporal demand, performance, effort, and frustration. In Fig. 5, we observe that along four of the six dimensions, differences between EyeO and the static control are statistically significant ($p < 0.05$). With EyeO, participants perceive reduced mental demand, improved performance, reduced effort, and reduced frustration. These results highlight an improved and more seamless user experience with autocalibration via EyeO compared to a static calibration approach. Qualitative feedback (Sec. 4.4.3) provides further support for these findings along the dimensions of mental comfort, performance, effort, and frustration.

We note that several participants asked us clarifying questions about the descriptions for physical demand and temporal demand. It is possible that the results for these two dimensions reflect discrepancies in their interpretations.

4.4.3 Overall Preferences. We asked participants "Which system do you prefer to use as an end-user of this device?". Their preferences between the systems were: EyeO 14 (73.68%), control 3 (15.79%), no preference 2 (10.53%). Open feedback from participants, summarized below, sheds light on these preferences.

Participants who preferred EyeO cited increased comfort and ease. One explained, *"I prefer [EyeO], because it was on average easier and required less mental and physical load."*. Another noted the accuracy of the system, *"As an end-user I prefer [EyeO] as it was more accurate for most of the sentences I typed. It was only frustrating for 2/5 sentences, as opposed to [the control] which was frustrating for most of the sentences."*. Participants also appreciated the real-time updates, one user noting *"[EyeO] adapted quickly to where I was looking"*. Others said, *"[EyeO] seemed to adjust to my typing and improve as I performed the task whereas [the control] was constantly bad (and while the consistency helped and I learned to adjust where the system faltered, I was frustrated I needed to apply additional effort to complete the task)"*; *"In [EyeO], I would start off having to recognize the offset and type accordingly, but after a few characters it would adapt and then I could actually look at the intended character, so it got progressively easier. In [the control] interface, when it worked (1 case) it was very smooth, but for the other 4 cases I had to identify the offset and consistently use it to type the whole phrase."* Some users also perceived that their own ability to type improved with EyeO, *"When starting with [EyeO], I did notice I was better at [gaze*



(a) Typing Speed (b) Number of Backspaces (c) Abort Frequency (d) Typing Speed (after EyeO's first read) (e) Number of Backspaces (after EyeO's first read)

Figure 4: Quantitative metrics of typing efficiency for the two systems. Error bars represent standard error (SE). Significance codes: * < .05, ** < .01, * <.001**

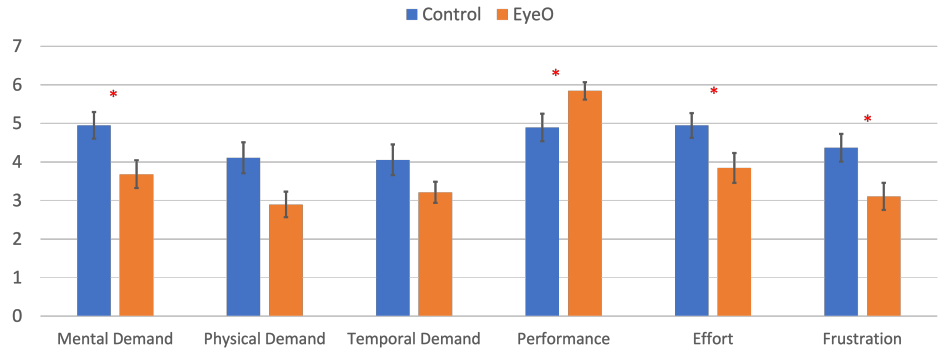


Figure 5: NASA-TLX responses about user workload for the two systems. Error bars represent standard error (SE). Significance codes: * < .05, ** < .01, * <.001**

typing] as I used more sentences, irrespective of how difficult the given sentence was to type as I was more comfortable overall.".

Since we purposefully introduced miscalibration to evaluate EyeO's corrections, some users also noted the struggle with EyeO before the autocalibration started with the first reading attempt, even though overall they preferred it over the static control. *"I found [the control] to be substantially more difficult to use. I felt significant strain while using it and felt noticeably tired after completing the five [control] tasks. [EyeO] was not easy to use...but I did not feel as strained while using it. Overall, I would describe [the control] as difficult to use and very uncomfortable, and [EyeO] as difficult to use but only somewhat uncomfortable".* When asked further to explain their reasoning for preferring [EyeO], they noted, *"I much preferred [EyeO]. I don't think I could have completed more than five sentences using [the control] without having to take a break. With [EyeO] I believe I could have done eight to ten sentences before needing a break. Even with a break, however, I do not think I could complete more than two five-sentence sessions using [the control] without needing to walk away from the computer for an extended (10+ minute) period of time. While answering the survey for [the control], I felt like I had a headache. I was worried that the headache would get worse during the next session of typing, but it did not. [EyeO] was not comfortable to use, but it did not cause me the sort of discomfort and mental strain I felt during and immediately after using [the control].".*

Participants who did not prefer EyeO cited some pitfalls of our approach. A few users preferred the reliability of errors in the static calibration system, despite additional cognitive load. As one participant explained, *"The error or offset between where I gaze and the detected cursor seems constant in [the control]. In [EyeO], the error is more random."* This reveals an opportunity for EyeO to be made more robust in terms of reducing the frequency of autocalibration updates as the session progresses. Additionally, we note that participants were neither made aware of the autocalibration feature nor its reliance on their verification of the last typed character. While most users fixated on the last typed character at least once during a trial, we noticed one user never verified what they typed with their gaze and instead relied on activated keys turning red (Fig. 2(c)). If a user does not look at the text box to verify the typed text, EyeO does not update the calibration. This contributed to unpredictability for a minority of users. Though the purpose of the study was to evaluate 'implicit' autocalibration, we believe if users were made aware of the underlying autocalibration technique and how it works, that could further enhance their experience.

One participant summarized the tradeoff between systems: *"It felt as if [EyeO] was adapting to the miscalibration in the eye tracker, whereas [the control] was not. The adaptability of [EyeO] has benefits and tradeoffs. It meant that [the control] was more predictable, whereas [EyeO] was less predictable. That said, there was perhaps less overall motor coordination effort involved in using [EyeO]"*. The same user noted, *"I think that overall I would prefer [EyeO] in the long-term."*

4.5 Verification of last typed character reading assumption

We analyze the data collected during our user study to validate the assumption used by EyeO (Algorithm 1), i.e. that participants fixate on the last typed character when they look up at the text box to verify what they typed recently (Fig. 7(b)). We measure the percentage of last character fixation attempts for all textbox lookups, lookups after typing each word, and lookups before or after a typing error (resulting in the use of the backspace key). To detect the last character fixation attempts, we rely on the gaze filtering procedure described in Appendix A.1.2 and the text box lookup criteria in Appendix A.1.3 and Algorithm 1. To validate that users often read the last character (making autocalibration with our proposed Algorithm 1 feasible), we analyze data from well-calibrated sessions (no induced miscalibration in any direction or the 0-miscalibration case described in Sec. 4.3) recorded with the control system during our user study. Using data solely from the control system for this analysis ensures that no gaze pointer/autocalibration updates occur throughout the typing session, preventing any conflation in our interpretation of fixation attempts. We can accurately compute the center location (x_c,y_c) of the last typed character displayed on the text box at the top of the screen. Fixations within the circular calibration zone of radius $\tau = 150$ pixels from the center (x_c,y_c) are considered as fixation attempts at the last-typed character. Characters on the textbox are 12-20 pixels wide and 25-30 pixels long.

We find that participants fixate on the last typed character for a large proportion of lookups to the text box at the top of the screen. Moreover, last character fixation attempts are more common after the completion of a word or when a user mis-types a character (i.e. uses the backspace key). Detailed results are shown in Table 2 averaging performance across participants. These findings confirm that a large fraction of all eye gaze reading attempts are comprised of fixations on the last typed character, validating our assumption. It further explains the effectiveness of such fixations to help autocalibrate the eye tracker in our proposed algorithm as evaluated in the user study (Fig. 4, Fig. 5). Thus, the results of our EyeO autocalibration approach are promising, indicating potential for broader adoption of future gaze typing systems.

Table 2: Validation of last character fixation assumption using the 0-miscalibration case in the control condition. Three types of lookup evaluations in which the participant verifies the last typed character are shown. Mean and standard error values depict the percentage of times a user fixates on the last character for each evaluation metric.

Overall lookups	After word completion	During typing errors
67.1% ± 2.6%	75.9% ± 4.2%	79.1% ± 4.0%

5 Discussion

As this work suggests, autocalibrating eye trackers without introducing additional tasks for the end user represents a step towards building more seamless user experiences. Gaze data offers vast potential as a form of input for computer interaction, effectively making technology more accessible and user-friendly [8, 10, 32, 34, 36]. With autocalibration, an array of seamless gaze interactions may become possible in natural settings, beyond controlled laboratory setups. To help inform such developments, below we discuss the limitations of our proposed technique (Sec. 5.1), implications of autocalibration for gaze typing (Sec. 5.2), and potential avenues for future work (Sec. 5.3).

5.1 Limitations

While our technique introduces new possibilities for making interactions more seamless, it also has several limitations. Firstly, our proposed gaze-typing autocalibration technique assumes that users at least fixate on the last typed character to verify what they typed. While we validate that this assumption holds for a majority of participant trials in our user study (Sec. 4.5 and Table 2), in rare scenarios, a user may never look at the text box to verify the typed text and instead rely on the visual changes during key selection (also noted in Sec. 4.4.3). Additionally, a user may read the entire word or larger parts of the typed text [27, 31]. In such cases, EyeO only detects fixations and autocalibrates when the user's gaze falls within the calibration zone (line 7 in Algorithm 1, Appendix A.1.3, Sec. 4.5) of the last-typed character, i.e. a circular zone with a small radius ($\tau = 150$ pixels) around the center of the last typed character. Systems that can accurately detect reading of different characters on the text box can potentially further improve the performance of EyeO.

Although we correct gaze offsets uniformly across the screen, prior work [11] has shown that eye-tracking accuracy can vary significantly across different screen regions. However, factors such as user movement or changes in the tracking setup, which introduce post-calibration errors, tend to cause miscalibration in consistent directions across the entire screen [11]. In such cases, correcting miscalibration based on fixations at the top of the screen can still improve typing speed in lower screen regions compared to leaving the miscalibration uncorrected. Moreover, using fixations from other application-specific gaze input targets available on different parts of the screen can help to more correctly adjust eye tracking offsets.

As in gaze tracking generally, our technique relies heavily on users having clear, unobstructed vision and the capacity to make smooth eye movements across the visual keyboard, which may not always be the case, particularly for users with specific ocular or neurological conditions. Additionally, eye movements during reading and writing are influenced by numerous cognitive and physiological factors, such as attention, fatigue, and cognitive load. These factors can cause dynamic changes in eye movement behavior, potentially affecting the computation of the miscalibration offset and hence the calibration accuracy. Also, note that in rare cases users might

use their peripheral vision to verify the typed characters. While traditional infrared eye trackers only track the center of the pupil and thus cannot directly detect peripheral gaze, there are indirect ways to measure such verification attempts in the form of quick saccades away from the typing fixations. However, without a clear detection of the gaze offset in the reading zone, there is no direct way to autocalibrate when users employ their peripheral vision.

5.2 Implications for Gaze Typing

We show that gaze offsets during verification of the last typed character can be leveraged for autocalibration of eye trackers and enhance gaze typing experience. Note that some prior works have proposed techniques beyond autocalibration such as dwell-time customization [29] and dynamic updates to keyboard parameters [3, 7, 9] that make gaze typing interactions adaptive and seamless. However, autocalibration is a complementary approach to improve the gaze typing experience and these additional approaches are beyond the scope of this work. Combining autocalibration with these other adaptive gaze typing techniques can further advance the goal of seamless gaze typing interfaces.

While we test EyeO with a single system setup, technically the algorithm is flexible enough to easily adapt to other monitor sizes, user distances, keyboard layouts, and PC eye tracking device manufacturers. However, such adaptation would require an initial phase of hyperaparameter tuning of the autocalibration parameters used in Algorithm 1. Additionally, to make EyeO's updates more accurate in different parts of the screen [9], using fixations on the center of a word during reading [31] and on other perception/reading targets in varied parts of the screen can make EyeO extend beyond one-point calibration.

5.3 Future Work

While we tested our autocalibration prototype with a single fixation point on the text box placed at the top of the screen, techniques that can leverage reading of longer typed text can provide more data about the gaze offset for autocalibration. Additional perception-based landmarks could be placed on different parts of the screen to help estimate a fine-grained miscalibration map and more accurate autocalibration. Evaluation of EyeO's adaptability to different monitor sizes, user distances, keyboard layouts, and eye tracking devices is an important direction for future work. Prior approaches using saliency maps, application-specific buttons, correlation of smooth pursuits and moving targets, next-word predictions, etc. can further augment EyeO for enhanced autocalibration, and in turn more seamless gaze interactions.

The autocalibration approach, by enhancing the accuracy and efficiency of eye tracking, can contribute towards creating more holistic, powerful, and intuitive interfaces, especially for real-world applications like augmented reality (AR) and virtual reality (VR). With increasing reliance on these technologies for various purposes – from gaming to professional training – the need for seamless and efficient interfaces is paramount. Our work can contribute towards this goal by combining eye gaze autocalibration updates with other natural interaction modalities (such as hand gestures, head movements, speech etc.), offering more accurate eye-tracking, and thereby improving the user's interaction with the AR/VR environment. One user study participant shared their excitement for integrating gaze typing technology with other modalities (e.g. speech or joystick input to smart TVs). They also recognized that the additional cognitive load may necessitate autocalibration and additional support for real-world deployment: *"As someone who is open to alternative means of typing due to physical restrictions, I like the potential of the technology but also acknowledge the potential strain for persons mentally and physically. It would be cool for persons typing on TVs though"*. Developing and studying how to best autocalibrate and support such multimodal interfaces is an exciting topic for future work.

We also note that while users with motor impairments (such as Amyotrophic Lateral Sclerosis or ALS) are early adopters of the gaze typing technology [9], we propose our autocalibration algorithm in an attempt to make the broader adoption of gaze typing interfaces more realistic. A user study to understand the specific impact of our autocalibration approach for accessibility through evaluation with ALS participants is also an important topic of future work.

6 Conclusion

Eye tracking technology has enabled computer interactions such as gaze typing for many users, with the potential to unlock additional future applications and interactions. However, the accuracy and efficiency of gaze tracking (including the application of gaze typing) can be significantly impacted by the calibration of the eye tracking system. Conventional calibration methods are time-consuming and require users to periodically perform manual calibrations, which can be inconvenient and disruptive to the user experience. Users also often compensate for miscalibration by intentionally offsetting their gaze, increasing their cognitive load, and confounding the automatic detection and correction of miscalibration. As a result, calibration problems remain a significant barrier to use [9].

To help address this problem for gaze typing, we provide the insight that users compensate their gaze during typing and not during reading. Thus, offsets detected during reading or verification attempts of typed characters can enable the correction of miscalibration. We demonstrate this approach through our EyeO prototype, which provides autocalibration for gaze typing to create a more seamless and accurate user experience. Our approach dynamically compensates for miscalibration when users read the text they have typed. Results from our user study suggest that by leveraging the natural gaze behavior of users during fixations on the typed text, our autocalibration system provides a more efficient, less mentally demanding, and overall preferable experience compared to traditional manual calibration. This opens up a novel adaptive approach for everyday users to interact with visual interfaces.

Acknowledgments

The authors would like to thank Pete Ansell, Jay Beavers, Jon Campbell, Susan Dumais, Harish Kulkarni, John Tang, and Shane Williams for helpful discussions and feedback.

References

[1] Yasmeen Abdrabou, Mariam Mostafa, Mohamed Khamis, and Amr Elmougy. 2019. Calibration-free text entry using smooth pursuit eye movements. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*. 1–5.

[2] Omair Shahzad Bhatti, Michael Barz, and Daniel Sonntag. 2021. EyeLogin-calibration-free authentication method for public displays using eye gaze. In *ACM Symposium on Eye Tracking Research and Applications*. 1–7.

[3] Xiuli Chen, Aditya Acharya, Antti Oulasvirta, and Andrew Howes. 2021. An adaptive model of gaze-based selection. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–11.

[4] Microsoft Corporation. 2018. Windows Eye Control. https://www.microsoft.com/en-us/garage/wall-of-fame/eye-control-windows-10/.

[5] Microsoft Corporation. 2021. MS Gaze Interaction Library. https://learn.microsoft.com/en-us/windows/communitytoolkit/gaze/gazeinteractionlibrary.

[6] Microsoft Corporation. 2021. Windows Community Toolkit. https://github.com/CommunityToolkit/WindowsCommunityToolkit/tree/rel/7.1.0/Microsoft.Toolkit.Uwp.Input.GazeInteraction.

[7] Wenzhe Cui, Rui Liu, Zhi Li, Yifan Wang, Andrew Wang, Xia Zhao, Sina Rashidian, Furqan Baig, IV Ramakrishnan, Fusheng Wang, et al. 2023. GlanceWriter: Writing Text by Glancing Over Letters with Gaze. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–13.

[8] Heiko Drewes. 2010. *Eye gaze tracking for human computer interaction*. Ph. D. Dissertation. lmu.

[9] Anna Maria Feit, Shane Williams, Arturo Toledo, Ann Paradiso, Harish Kulkarni, Shaun Kane, and Meredith Ringel Morris. 2017. Toward everyday gaze input: Accuracy and precision of eye tracking and implications for design. In *Proceedings of the 2017 Chi conference on human factors in computing systems*. 1118–1130.

[10] Jensen Gao, Siddharth Reddy, Glen Berseth, Nicholas Hardy, Nikhilesh Natraj, Karunesh Ganguly, Anca D Dragan, and Sergey Levine. 2022. X2T: Training an x-to-text typing interface with online learning from user feedback. *arXiv preprint arXiv:2203.02072* (2022).

[11] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[12] Mamoru Hiroe, Michiya Yamamoto, and Takashi Nagamatsu. 2018. Implicit user calibration for gaze-tracking systems using an averaged saliency map around the optical axis of the eye. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. 1–5.

[13] Mamoru Hiroe, Michiya Yamamoto, and Takashi Nagamatsu. 2023. Implicit User Calibration for Gaze-tracking Systems Using Saliency Maps Filtered by Eye Movements. In *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications*. 1–5.

[14] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. 2011. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford.

[15] Kiyoshi Hoshino, Yuki Noguchi, and Yuya Nakai. 2020. Gaze estimation with easy calibration method. In *Proceedings of the 2020 5th International Conference on Intelligent Information Technology*. 102–106.

[16] Michael Xuelin Huang, Tiffany CK Kwok, Grace Ngai, Stephen CF Chan, and Hong Va Leong. 2016. Building a personalized, auto-calibrating eye tracker from user interactions. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 5169–5179.

[17] Thomas E Hutchinson, K Preston White, Worthy N Martin, Kelly C Reichert, and Lisa A Frey. 1989. Human-computer interaction using eye-gaze input. *IEEE Transactions on systems, man, and cybernetics* 19, 6 (1989), 1527–1534.

[18] Robert JK Jacob and Keith S Karn. 2003. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In *The mind's eye*. Elsevier, 573–605.

[19] Pawel Kasprowski and Katarzyna Harezlak. 2016. Implicit calibration using predicted gaze targets. In *Proceedings of the ninth Biennial ACM symposium on eye tracking research & applications*. 245–248.

[20] Pawel Kasprowski and Katarzyna Harezlak. 2018. Comparison of mapping algorithms for implicit calibration using probable fixation targets. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. 1–8.

[21] Mohamed Khamis, Ozan Saltuk, Alina Hang, Katharina Stolz, Andreas Bulling, and Florian Alt. 2016. TextPursuits: using text for pursuits-based interaction and calibration on public displays. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 274–285.

[22] Otto Hans-Martin Lutz, Antje Christine Venjakob, and Stefan Ruff. 2015. SMOOVS: Towards calibration-free text entry by gaze using smooth pursuit movements. *Journal of Eye Movement Research* 8, 1 (2015).

[23] I Scott MacKenzie and R William Soukoreff. 2002. Text entry for mobile computing: Models and methods, theory and practice. *Human–Computer Interaction* 17, 2-3 (2002), 147–198.

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

[25] 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*. 15–22.

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

[27] George W McConkie, Paul W Kerr, Michael D Reddix, and David Zola. 1988. Eye movement control during reading: I. The location of initial eye fixations on words. *Vision research* 28, 10 (1988), 1107–1118.

[28] Carlos H Morimoto and Marcio RM Mimica. 2005. Eye gaze tracking techniques for interactive applications. *Computer vision and image understanding* 98, 1 (2005), 4–24.

[29] Martez E Mott, Mathieu Nancel, Gierad Laput Kumar, Chris Harrison, and Antti Oulasvirta. 2017. Cascading gaze dwells: on the efficiency of bottom-up activation in gaze-based cascading menus. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 5308–5319.

[30] Ken Pfeuffer, Melodie Vidal, Jayson Turner, Andreas Bulling, and Hans Gellersen. 2013. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 261–270.

[31] Keith Rayner. 1979. Eye guidance in reading: Fixation locations within words. *Perception* 8, 1 (1979), 21–30.

[32] Kerstin Ruhland, Christopher E Peters, Sean Andrist, Jeremy B Badler, Norman I Badler, Michael Gleicher, Bilge Mutlu, and Rachel McDonnell. 2015. A review of eye gaze in virtual agents, social robotics and hci: Behaviour generation, user interaction and perception. In *Computer graphics forum*, Vol. 34. Wiley Online Library, 299–326.

[33] 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.

[34] Akanksha Saran, Srinjoy Majumdar, Elaine Schaertl Short, Andrea Thomaz, and Scott Niekum. 2018. Human gaze following for human-robot interaction. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8615–8621.

[35] Akanksha Saran, Elaine Schaertl Short, Andrea Thomaz, and Scott Niekum. 2020. Understanding teacher gaze patterns for robot learning. In *Conference on Robot Learning*. PMLR, 1247–1258.

[36] Akanksha Saran, Ruohan Zhang, Elaine Schaertl Short, and Scott Niekum. 2020. Efficiently guiding imitation learning agents with human gaze. *arXiv preprint arXiv:2002.12500* (2020).

[37] Shreshth Saxena, Elke Lange, and Lauren Fink. 2022. Towards efficient calibration for webcam eye-tracking in online experiments. In *2022 Symposium on Eye Tracking Research and Applications*. 1–7.

[38] Oleg Špakov, Howell Istance, Tiia Viitanen, Harri Siirtola, and Kari-Jouko Räihä. 2018. Enabling unsupervised eye tracker calibration by school children through games. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. 1–9.

[39] Oleg Špakov, Harri Siirtola, Howell Istance, and Räihä Kari-Jouko. 2017. Visualizing the reading activity of people learning to read. *Journal of Eye Movement Research* 10, 5 (2017).

[40] Kang Wang, Shen Wang, and Qiang Ji. 2016. Deep eye fixation map learning for calibration-free eye gaze tracking. In *Proceedings of the ninth biennial ACM symposium on eye tracking research & applications*. 47–55.

[41] Ruohan Zhang, Akanksha Saran, Bo Liu, Yifeng Zhu, Sihang Guo, Scott Niekum, Dana Ballard, and Mary Hayhoe. 2020. Human gaze assisted artificial intelligence: A review. In *IJCAI: Proceedings of the Conference*, Vol. 2020. 4951.

[42] Xiaoyi Zhang, Harish Kulkarni, and Meredith Ringel Morris. 2017. Smartphone-based gaze gesture communication for people with motor disabilities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2878–2889.

[43] Yunfeng Zhang and Anthony J Hornof. 2014. Easy post-hoc spatial recalibration of eye tracking data. In *Proceedings of the symposium on eye tracking research and applications*. 95–98.

A Appendix

A.1 Additional Details on the EyeO Prototype

A.1.1 System Setup. Currently available visual keyboards, such as in Windows Eye Control [4] (Fig. 1(a)) are explicitly calibrated using a 9-point calibration method (Fig. 6). However, EyeO provides more control to adaptively process and update the miscalibrated gaze coordinates in real-time for such keyboards.

Figure 2 illustrates the mechanism for typing a character on the visual keyboard we use in this work. A user's gaze location as detected by the eye tracker is displayed as a red dot on the screen. If they fixate on a key for 50 ms, the system initiates a dwell timer of 400ms. The start of the timer is depicted by a green rectangle on the selected key (Fig. 2(a)). During fixation on the key, the green rectangle slowly decreases in area, eventually collapsing at the center of the key (Fig. 2(b)). When the timer finishes, the user receives visual feedback that the character on the key has been typed: the key turns red (Fig. 2(c)) and the letter is added to the text box at the top of the screen (Fig. 2(d)).

The tracked gaze is directed toward the full screen visual keyboard (Fig. 2), displayed on a 14-inch Lenovo laptop screen. The visual keyboard can adapt to any screen size. While our autocalibration algorithm is tested on this specific keyboard layout, it is not restrictive and can be extended to work with any other keyboard layout by tuning the autocalibration hyperparameters used in Algorithm 1. The keyboard implementation we used is available via the open-source Windows community toolkit [6] and our autocalibration algorithm is added to the keyboard functionality via a UWP application available as part of Microsoft's open-source Gaze Interaction Library [5]. The EyeO prototype was programmed using the Tobii Pro SDK (designed to offer access to gaze data from Tobii eye trackers) in C#. It facilitates the capture and processing of x, y coordinates from the eye tracker, providing necessary data for EyeO to leverage and update the gaze coordinates in real time. The software for our autocalibration algorithm was developed in Python, which received the 2D gaze coordinates in real-time from the UWP application via Google's remote procedure call (RPC) protocol. This allowed Python scripts to access gaze data, analyze it, and apply the necessary miscalibration corrections to display back on the visual keyboard application in real-time (~ 60 Hz). The system was tested on a laptop with an Intel Core i7–10610U and 16 GB memory.

A.1.2 Gaze Filtering. Eye gaze movements can be characterized as: (a) fixations, (b) saccades, (c) smooth pursuits, and (d) vestibulo-ocular movements [14, 35]. Visual fixations maintain the focus of gaze on a single location. Fixation duration varies based on the task, but one fixation is typically 100-500 ms, although it can be as short as 30 ms. Saccades are rapid, ballistic, voluntary eye movements (usually between 20-200 ms) that abruptly change the point of fixation. Smooth pursuit movements are slower tracking movements of the eyes that keep a moving stimulus on the fovea. Such movements are voluntary in that the observer can choose to track a moving stimulus, but only highly trained people can make smooth pursuit movements without a target to follow. The gaze typing interface used in this work does not consist of any moving targets. Vestibulo-ocular movements stabilize the eyes relative to the external world to compensate for head movements. These reflex responses prevent visual images from slipping on the surface of the retina as head position changes. With respect to gaze typing on visual keyboards, we don't expect users to move their heads significantly.

In our setup, we expect users to primarily use fixations and saccades during gaze typing [9] as smooth pursuits and vestibulo-ocular movements are not present in our trials. To distinguish fixations from saccades, we consider spatial and temporal criteria from the taxonomy of fixation identification algorithms described by Salvucci and Goldberg [33]. In our work, we use the IVDT algorithm from the taxonomy which uses velocity-based (threshold $50°/s$) and area-based (threshold $2°$ of visual angle) criteria as spatial characteristics and duration-based criteria as a temporal characteristic to filter out fixations from saccades. We first filter out eye movements with very high speeds (a large distance traversed over a very short period of time is likely a saccade). If gaze continues to not be classified as a saccade for more than 100 ms while falling in a region of interest, then we consider it a fixation.

A.1.3 Autocalibration Algorithm. The algorithm input comprises of a stream of coordinates (x_t, y_t) from the device driver, as well as several variables describing the state of the keyboard interface. It produces estimated gaze coordinates $(\widehat{x}_t, \widehat{y}_t)$ based on adaptively estimated calibration errors $(\epsilon_{x_t}, \epsilon_{y_t})$, which are refined whenever the user is inferred to be reading the last typed character displayed in the text box. The difference between the center location of the last typed character in the text box and the predicted gaze location during a verification attempt is the estimated amount of miscalibration at any given moment. This estimate is accumulated over time to give a smoothed autocalibration update for the predicted gaze location.

The user's gaze is assumed to be directed towards the text box when it falls above the keyboard layout and is assumed to be fixated on verifying the last typed character when it falls within a certain threshold distance τ (calibration zone) from the center of the last typed character. We first filter out saccades and then detect fixations (Sec. A.1.2) within the calibration zone (lines 5-7 in Algorithm 1). Visual fixations maintain the focus of gaze on a single location and are potential candidates for verification reading attempts of the last typed character.

We assume that the user fixates on the center of the last typed character when they look up to verify what they typed with their gaze. We detect the offset in calibration based on how far their gaze is predicted by the eye tracker and where the center of the last typed character is on the screen (lines 8-10 in Algorithm 1). Our system computes a moving average of the calibration offsets (line 9), which enables the calibration to be updated continuously and smoothly in real-time. The system continuously updates the calibration error based on the user's gaze behavior during verification of the typed text. The computed correction for the detected miscalibration is continuously applied even when the user's gaze moves away from the text box (i.e. while typing on the visual keyboard) as depicted in lines 12-15. Our simplified algorithm and prototype implementation allow for this adaptive interface to correct for miscalibration in real-time. The corrected gaze coordinates are displayed to the user with the updated location of the red gaze cursor (Fig. 2(a)).

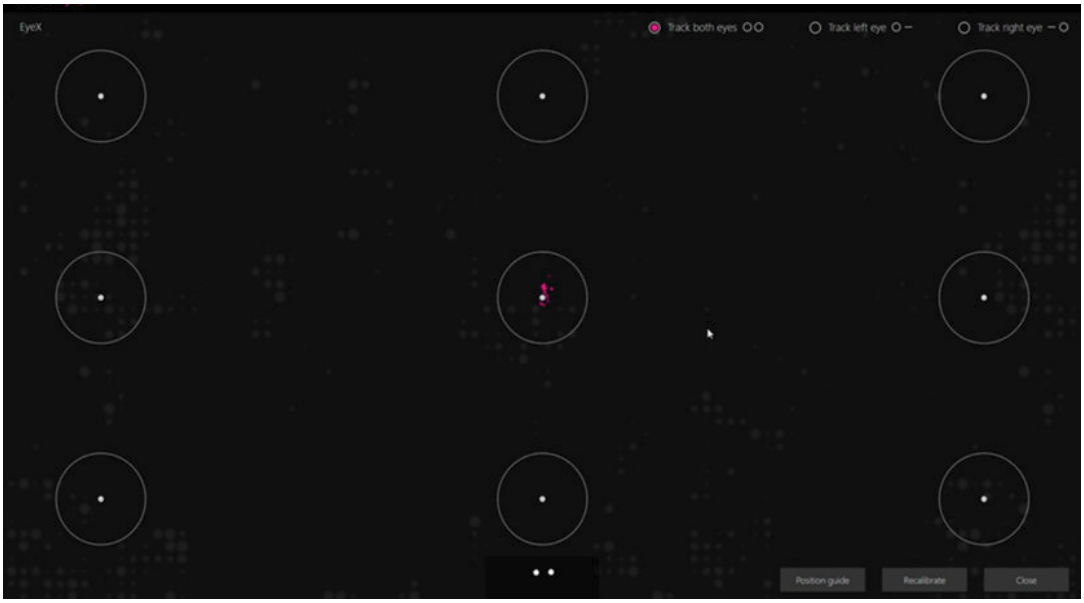


Figure 6: Traditional 9-point calibration system used to calibrate eye tracking hardware.

We instantiate the proposed algorithm with the calibration zone size parameter (the distance around the location of the last typed character where we detect gaze fixations for reading) $\tau = 150$ pixels, window size (to compute the running average) $w = 64$, and calibration error bound (the maximum amount of miscalibration tolerated/corrected by the system) $b = 200$ pixels. These parameter values were determined during early prototype testing based on: (1) maximum limit of miscalibration amounts under which gaze typing could be successfully performed with user compensation and (2) desirable rate of calibration updates that minimized latency and interruptions to real-time interaction frequency.

A.2 Additional Details on the User Study

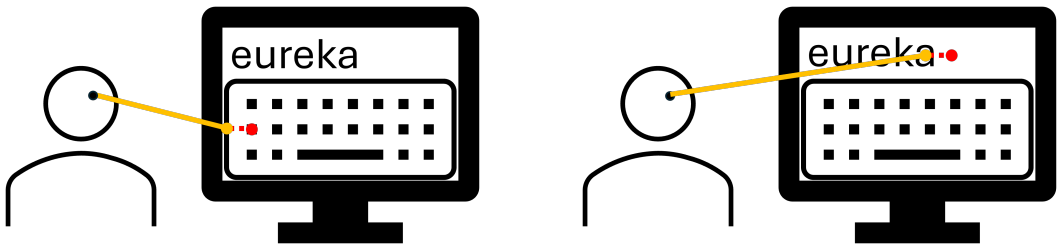
A.2.1 System Setup. Both systems (EyeO and control) shared the same visual keyboard layout and gaze typing application. The only difference was that EyeO contained an additional layer of Python software to detect and correct for miscalibration (described in Section 3.2). We used a remote video-based tracker (typically attached to the bottom of a computer screen). It captures the gaze location of the user in screen coordinates at regular intervals (recording frequency $\sim$ 60Hz). It is a head-pose-free tracker, that can tolerate minor head movements within a certain range of locations and orientations. All manual calibrations were performed via the standard Tobii SDK 9-point calibration procedure (Fig. 6). We note that this calibration procedure is run via a proprietary black-box software module. End-users and application designers do not have any flexibility or control over this module. Our proposed approach adds a layer of control on top of this black-box software module to correct users' gaze coordinates under miscalibration. Importantly, our proposed technique is flexible to augment any other existing proprietary or open-source calibration software stack.

A.2.2 Procedure. Participants were initially introduced to gaze typing and given a brief tutorial on how to use the on-screen keyboard. The experiment consisted of an initial calibration procedure, a practice round, and trials of both EyeO and the control (within-subjects study). The practice round was attempted with the well-calibrated control system during which they typed a couple of simple phrases ('happy new year', 'hello world') in the gaze typing application. During the system trials, the participants were not told which system they were testing (EyeO or control). The systems were anonymized as system A and system B respectively. The order of the EyeO and control phases was counterbalanced across participants.

Users were free to abort a typing session if they found the setup too cumbersome to continue with the task. This was especially true for some users who experienced fatigue or found it very challenging to type under induced miscalibration. Any such partial typing sessions that were aborted are reflected in our results in Sec. 4.4.1.

A.2.3 Miscalibration parameters. Miscalibration of different amounts was purposefully introduced for each system (EyeO and Control) to evaluate the typing experience of users (Sec. 4.3). We do this for two reasons: (1) to accurately measure and report quantitative performance improvements with autocalibration. (2) stress test the gaze typing systems in addition to the miscalibrations that naturally appear over time with user fatigue, lighting changes, head or body movement etc. Participants typed a set of unique phrases (one each for the different miscalibration amounts) under the two gaze typing systems. No phrase was repeated across participants or across the two systems. The users were made aware that the systems would exhibit miscalibration during the study, but were not made aware of any differences between the two systems.

The gaze tracker was calibrated at the beginning of each of the two system trials (EyeO and Control). Given that users viewed the



(a) Gaze used for typing: the user compensates for miscalibration to the right (depicted by the dotted line between the red gaze prediction and the user's intended fixation location in yellow), intentionally focusing to the left (yellow) of the key they want to select (the first key on the second row of the keyboard).

(b) Gaze used for reading: the user does not compensate for miscalibration (predicted gaze location in red away from the letter being read), looking directly at the text they read (intended fixation location in yellow actually being read by the user). Their gaze does not control the system, so they are free to look where is natural. The difference between the red and yellow locations provides a signal to correct for miscalibration.

Figure 7: Demonstration of the difference in gaze offset between typing and reading. We leverage this difference to detect and correct miscalibrations, providing the user with a more seamless experience.

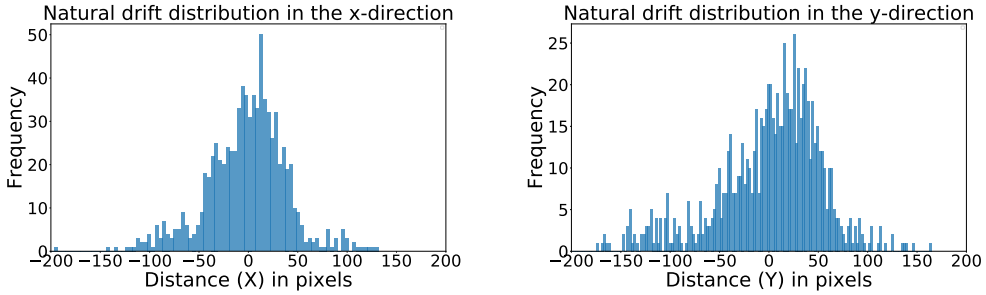


Figure 8: Distribution of natural drift in the accuracy of the eye tracker in the x and y directions.

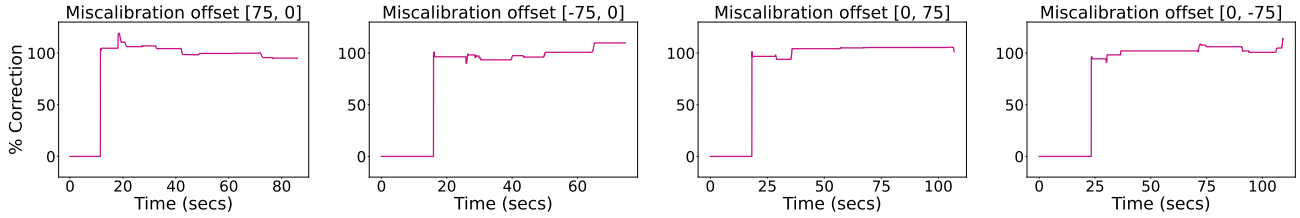


Figure 9: Autocalibration updates over time by EyeO for a representative participant of the user study. The denominator for the y-axis represents the miscalibration amount introduced in the experiment and the numerator represents the amount of miscalibration corrected via autocalibration.

screen at a distance of approximately 75 cm, an induced miscalibration of 75 pixels corresponds to ~ 0.9 degrees of visual angle. Since we do not restrict users's seating in any way, we expect the miscalibration to lie between ~ 0.7-1.2 degrees of visual angle. Errors were introduced independently in the x and y directions to better isolate the impact of compensation in the two directions, following the study design of Feit et al. [9]. The induced miscalibrations affected the entire screen uniformly. While prior work [9] have shown that eye tracking accuracy varies significantly across different regions of the screen (with worse performance in the bottom right of the screen versus the top of the screen), several factors that introduce errors after good calibration (such as movement of the user or tracking setup) will result in miscalibration in the same direction on all parts of the screen [9]. In such cases even correcting for miscalibration via fixations at the top of the screen will be beneficial for typing faster in the lower regions of the screen compared to not correcting for any miscalibration at all. Based on different application needs

and interface designs, gaze input signals from diverse regions of the screen can help with more accurate autocalibration.

The miscalibration parameters were determined during early prototype testing. The values were chosen such that users could face a significant challenge during the typing task while still being able to complete it by compensating. Additionally, we also measure the degradation in the accuracy of the eye tracker caused by natural usage. We ask 10 separate users to calibrate their eyes with the tracker, move away for a short walk, and then return to type a test phrase from a standard corpus [23]. Fig. 8 shows that the drift distribution in both the x and y directions primarily lies in the range of −50 to 50 pixels. Thus our induced miscalibration parameter of 75 pixels is towards the tail end of the distribution and serves as a way to stress test the system in more extreme conditions than those observed on average.

A.2.4 Additional Results. To further illustrate the effectiveness of EyeO, we show how autocalibration corrections update over time during individual typing sessions for a single representative participant (Fig. 9). For miscalibration induced in different directions, we find that EyeO adapts to the error nearly perfectly. The sharp jump closing the error gap shown in Fig. 9 indicates how a single last-character reading attempt early in the session (usually within $\sim$ 25 seconds for the representative user) can be sufficient to ease the burden of miscalibration. Note the control system never updates during the session — the user thus must continuously compensate to successfully type under miscalibration with such a system.