UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

HexGaze: Hexagonal Virtual Keyboard with Word Prediction for Eye Typing

Robert Filipe Susloparov Cachapa

Mestrado em Engenharia Informática

Dissertação orientada por:
Prof. Doutor Manuel João Caneira Monteiro da Fonseca

2025

To my family and friends

Acknowledgments

This marks the end of an academic journey, one that would not have been possible without the support and guidance of many people, to whom I am deeply grateful.

First, I would like to thank Professor Manuel João Fonseca for his guidance throughout this thesis and for his support in the design and development of our solution. His ideas, knowledge, and expertise were invaluable in shaping this work and bringing it to the final form presented here.

I am also grateful to all participants who contributed their time and patience to the HexGaze testing process, which was fundamental to the success of this study.

Finally, I would like to express my gratitude to my family for their hard work and for giving me the opportunity to pursue this path, as well as to my friends and girlfriend, who stood by me and supported me throughout this journey. This work would not have been possible without them.

Resumo

A escrita ocular, ou *eye-typing*, é uma forma de introdução de texto que permite ao utilizador escrever apenas com os movimentos dos olhos, recorrendo a um sistema de eye tracking que interpreta o ponto de fixação e o transforma em coordenadas no ecrã. Dependendo de como essas coordenadas são processadas por técnicas especiais, podem ser refletidas em ações no ecrã, como escrever uma letra ou selecionar um objeto. Desta forma, estas técnicas de seleção permitem ao utilizador escrever com o olhar. Esta forma de interação é especialmente valiosa em situações onde o uso de interfaces tradicionais, como teclado e rato, não é possível, sendo particularmente útil quando as pessoas têm as mãos ocupadas, em sistemas de realidade virtual ou para pessoas com limitações físicas que dependem deste tipo de tecnologia para comunicar e operar um computador.

No entanto, a maioria das técnicas de escrita ocular existentes ainda não consegue atingir um bom equilíbrio entre velocidade, precisão e conforto de utilização. Apesar do elevado volume de investigação na área, com mais de 4.700 publicações identificadas em 2024 [16], ainda não surgiu uma técnica amplamente adotada. Tal deve-se, em grande parte, a limitações persistentes dos sistemas baseados no olhar, nomeadamente o desconforto e a fadiga acumulados durante utilizações prolongadas, bem como a necessidade de realizar movimentos oculares bruscos e repetitivos.

Outro problema relevante são as elevadas taxas de erro, frequentemente causadas pela baixa precisão dos eye trackers. Esta limitação pode levar à seleção de letras adjacentes à pretendida, mesmo quando o utilizador não fixou diretamente o olhar nessas teclas. Acresce ainda o problema do "Toque de Midas", no qual ocorrem seleções involuntárias enquanto o utilizador observa o teclado à procura da letra desejada. Este fenómeno resulta do facto de o olhar ser, naturalmente, o principal meio de recolha de informação visual e não de introdução de dados. Assim torna-se essencial o desenvolvimento de técnicas que permitam visualizar claramente a letra pretendida sem a ativar acidentalmente. Por último, temos a velocidade de escrita que constitui outro fator crítico.

Para mitigar os problemas anteriormente referidos, muitas técnicas recorrem a mecanismos de seleção com tempos de espera (dwell time), nas quais o utilizador tem de manter o olhar fixo numa tecla durante um determinado período (tipicamente entre 600 ms e 1000 ms), ou a movimentos oculares específicos. Contudo, como o utilizador tem de observar e selecionar simultaneamente com os olhos, estas abordagens acabam por limitar a velocidade de escrita pois consomem tempo, tornando-a significativamente inferior à de um teclado físico tradicional.

Com este projeto, definimos como objetivos o desenvolvimento de uma técnica de escrita ocular confortável, rápida e de baixo custo, capaz de funcionar mesmo com eye trackers imprecisos

e de permitir a escrita sem causar esforço significativo. Para compreender como atingir estes objetivos, começámos por estudar as alternativas existentes, identificando os aspetos em que apresentavam maior sucesso, de modo a aproveitar essas componentes, bem como os pontos em que essas técnicas falham, para evitar cometer os mesmos erros.

A nossa análise do desempenho das técnicas dividiu-se em três aspetos: i) Palavras por minuto (Words Per Minute – WPM), que indica quantas palavras é possível escrever por minuto com a técnica, considerando que cada palavra corresponde, normalmente, a uma sequência de cinco caracteres; quanto mais elevado o valor, melhor, pois significa que o utilizador escreveu mais palavras num curto período de tempo; ii) Teclas por carácter (Keystrokes Per Character – KSPC), que indica quantas teclas o utilizador precisou de selecionar com os olhos para escrever cada letra, sendo que valores mais baixos representam maior eficiência, uma vez que são necessárias menos seleções para escrever a frase; iii) Distância mínima da frase (Minimum String Distance – MSD), uma métrica que quantifica a taxa de erro ao determinar o número mínimo de operações de apagar, inserir ou substituir necessárias para transformar a frase escrita na frase pretendida. Estudámos ainda o funcionamento dos eye trackers, de forma a criar uma técnica capaz de ser utilizada eficazmente mesmo em dispositivos de baixa precisão.

Neste trabalho, criámos um teclado virtual dedicado e otimizado para seleções eficientes e precisas com equipamento de baixo custo, confortável e adaptado às preferências dos utilizadores, denominado HexGaze. O HexGaze é uma técnica de introdução de texto composta por um teclado hexagonal, com uma interface de layout claro, simétrico e intuitivo, dividido em duas secções principais: um painel de estado no topo e um canvas interativo com o teclado virtual no centro. O núcleo da aplicação é o teclado dinâmico, construído a partir de uma rede interligada de hexágonos. Optámos por este layout porque os hexágonos se conectam naturalmente, preservam linearidade e direção, são compactos e ergonómicos, e a sua divisão em seis segmentos permite agrupar letras e símbolos de forma eficiente. Cada hexágono contém seis segmentos com conjuntos de seis letras coloridas, agrupadas alfabeticamente. O layout inclui também teclas de ação dedicadas, como apagar a última letra, apagar a última palavra e a barra de espaço.

O HexGaze combina dois métodos de seleção: Gestos e Dwell Time. A seleção por gestos é o método principal, sendo otimizada para velocidade e fluidez. O utilizador seleciona uma letra fixando o olhar no segmento do hexágono de grupo (hexágono que contém os agrupamentos de seis letras em cada segmento), o que faz surgir um hexágono secundário com as letras do segmento previamente escolhido, distribuídas individualmente pelos seus segmentos. Ao olhar para este novo hexágono de letras, este é "escolhido" e posicionado no centro do ecrã. Este processo repete-se novamente para o hexágono de letras, porém, desta vez, ao olhar para o hexágono adjacente ao segmento que contém a letra pretendida, essa letra é selecionada. De seguida, é emitido um som de confirmação e a interação regressa ao hexágono de grupo. A animação utilizada minimiza movimentos bruscos e mantém uma elevada taxa de frames, contribuindo para a redução da fadiga ocular.

Caso o utilizador não queira realizar todos estes movimentos e prefira uma interação mais

confortável em troca de alguma velocidade, pode também selecionar diretamente o segmento da letra no hexágono de letras utilizando Dwell Time. O modo de Dwell Time funciona como um mecanismo de apoio, exigindo que o olhar se mantenha sobre a letra ou palavra durante um intervalo de tempo específico (500 ms para letras e 700 ms para palavras). A interface inclui ainda mecanismos de feedback visual e auditivo, bem como um campo de texto que apresenta, em tempo real, as letras digitadas pelo utilizador.

O mecanismo de sugestões utiliza dois motores: o de previsão da palavra atual, baseado em unigramas, que apresenta as seis opções mais prováveis com base no prefixo escrito, e o de previsão da próxima palavra, baseado em bigramas, que sugere as palavras subsequentes mais prováveis à palavra já escrita.

Realizámos dois estudos para avaliar o desempenho da nossa técnica. No primeiro estudo, comparámo-la com outras duas técnicas de introdução de texto ocular, de forma a analisar o desempenho nas métricas referidas anteriormente. Os testes foram realizados com 21 participantes, cada um numa única sessão composta por três partes, nas quais os participantes escreveram oito frases com cada uma das técnicas. No final de cada parte, os utilizadores responderam a um questionário de feedback sobre a técnica utilizada. No segundo estudo, analisámos como é que o uso da técnica evolui ao longo do tempo, de modo a avaliar quão rapidamente e quão bem os utilizadores conseguiam melhorar após várias sessões de escrita. Os testes foram realizados com 11 participantes, cada um dos quais realizou oito sessões (duas por dia, durante quatro dias), escrevendo oito frases em cada sessão. Após cada sessão, os utilizadores responderam a um questionário de feedback sobre a técnica.

No estudo I os resultados mostraram que o HexGaze superou as restantes técnicas em velocidade de escrita (WPM), taxa de erro (MSD) e eficiência (KSPC). O feedback dos participantes indicou ainda que o HexGaze foi percebido como a técnica mais confortável, menos fatigante e mais fácil de aprender.

No estudo II os resultados revelaram um forte efeito de aprendizagem onde a velocidade de escrita praticamente duplicou da primeira para a oitava sessão e a eficiência das teclas melhorou significativamente, enquanto as taxas de erro se mantiveram estáveis e baixas. Os participantes não relataram aumento de fadiga ocular e o comforto foi aumentando em cada sessão, o que demonstra que a técnica é sustentável para utilizações prolongadas e de maior comforto quanto melhor for a experiência do utilizador.

A análise global destes estudos mostra que o HexGaze oferece um bom equilíbrio entre velocidade, precisão e conforto, apresentando também uma curva de aprendizagem bastante promissora e sendo fácil de aprender para principiantes, mostrando ser uma técnica promissora para escrita ocular.

Palavras-chave: Escrita Ocular, HexGaze, Dwell Time, Eye tracker, Introdução de texto

Abstract

Although eye typing has existed for several years, many gaze-based systems still struggle when using low-cost eye trackers, resulting in slow typing speeds, high error rates, and increased user fatigue and discomfort. These challenges highlight the need for continued improvement and innovation in eye typing solutions. This work introduces and evaluates HexGaze, a gaze-based text entry technique designed to be comfortable, fast, and tolerant of low-cost, imprecise eye trackers. The system features a unique virtual keyboard with a hexagonal layout that keeps user interaction centered on the screen, where eye trackers accuracy is the highest. It combines fluid, gesture-based selection with a dwell-time method to allow users the freedom to chose the most comfortable approach. It also integrates a word prediction system for both current-word completion and next-word suggestions to enhance typing efficiency.

Two user studies were conducted to evaluate HexGaze. The first study compared it against two existing techniques, WordPop and Dasher. The results showed that HexGaze achieved higher typing speeds (5.96 WPM), lower error rates, and greater keystroke efficiency than WordPop. Although HexGaze also outperformed Dasher, the difference was not statistically significant, suggesting a trend in favour of HexGaze but not a conclusive advantage. Participants also rated it as the most comfortable, least fatiguing, and easiest to learn. The second study analysed the learning curve over eight sessions, revealing a strong and consistent improvement in performance. Typing speed nearly doubled, from an average of 5.59 WPM (first session) to 9.94 WPM (last session). Efficiency (KSPC) also increased substantially from the first session to the last. Crucially, error rates remained stable and eye fatigue remained low throughout, demonstrating the technique's suitability for prolonged use.

Finally, the findings demonstrate that HexGaze offers a superior balance of speed, accuracy, and user comfort. Its intuitive design and strong learning curve make it a promising, user friendly, and practical solution for eye typing.

Keywords: Eye Typing, HexGaze, Text entry, Gestures, Dwell Time

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we cover the motivation for our work as well as the main goals we want to achieve. Finally we give an insight of the developed solution and describe how this document is organized.

1.1 Motivation

Nowadays, there are many ways to interact with a computer, including keyboards, mice, and touch screens. Eye typing is one such interaction method and was primarily developed to support hands-free use and to enable computer access for people with disabilities for whom traditional input methods are not viable due to physical limitations. In 2021, approximately 1.3 billion people were living with some form of disability, and among these, about 2–4% experienced significant functional challenges [32]. For these users who require hands-free interaction or who possess functional disabilities, the ability to interact with a computer using eye movements captured through an eye tracker can be essential. To make this possible, specialized eye-typing techniques are required, allowing gaze to function as an input method similar to a keyboard, enabling users to type text and select interface elements. Although many eye-tracking techniques have been developed, almost none has yet provided the perfect solution. Gaze typing remains challenging because the eyes are primarily used to gather visual information rather than to perform control functions. As a result, a gaze-based system cannot simply assume that users intend to select every object they look at, as this would lead to frequent unintentional selections. When combined with the limitations of low-cost eye trackers, this issue often results in involuntary letter inputs. Achieving a balance between speed, comfort, and low error rates while addressing these challenges is particularly difficult. In 2024, a study identified approximately 4,705 papers related to eye typing and gaze-based communication, enveloping both novel techniques and improvements to existing methods [16]. Despite this extensive body of research, a widely adopted or "mainstream" eye typing technique has yet to emerge. This is largely due to persistent challenges and limitations associated with gaze-based systems. As these technologies are not yet integrated into everyday use, they remain in development, with significant room for refinement before they can be considered practical for the average user. Moreover, many existing techniques, although effective and fast, often suffer from issues such as discomfort, monotony, or user fatigue during extended sessions.

1.2 Goals

In this work our goal is to develop a comfortable, fast and low-cost imprecise eye tracker tolerant gaze-based typing technique that surpasses the existing ones, allowing users to type without feeling eye strain, and maximizing comfort. Additionally, we want to create a dedicated virtual keyboard that will allow users to efficiently and accurately select keys using a low-cost eye-tracking device. To accomplish this, we must first evaluate other similar techniques, analyse their strengths and weaknesses to understand why they succeed or fail. Given that eye trackers play a big role in this work, it is also essential to understand their technology and limitations, so we manage them effectively. Finally, we intend to compare our technique with others, to assess how comfortable and fast our method is and see if our goals are achieved. We also want to study the learning curve of users by having them use our solution during a week with daily sessions.

1.3 Developed Solution

The proposed eye typing technique introduces a boundless open canvas featuring a central hexagon divided into six segments, each holding six color-coded letters. When a user gazes at a segment, a secondary hexagon appears next to the selected segment, presenting the six corresponding letters, each retaining its color-coded background to assist in peripheral recognition. As users continue typing, each selected letter spawns a new hexagon in a chain-like layout, allowing continuous input without returning to the start. We incorporated word prediction methods and visual and auditory feedback to enhance speed and ease of use, making it easier for users to adapt quickly and minimize error rates without compromising on comfort and fluidity. The system uses dynamic word prediction, filtering out invalid letters and replacing them with predicted completions to guide the user efficiently. A unique “beep” sound provides auditory feedback to reassure the user of letter selection, and a progress bar provides visual feedback. The virtual keyboard offers two typing alternatives, one using Dwell and the other Dwell Free. In the latter selection method the user can select letters without having to gaze at the letter and wait a certain time to select. In the Dwell Time option, users have to gaze at the letter segments or predicted words for a certain period of time to select them and restart the chain. Finally, we designed the virtual keyboard to always be around the center of the screen, since this is where the low-cost eye trackers are more precise.

1.4 Document Organization

This document is organized as follows. In Chapter 2, we provide the background of our research, addressing the challenges associated with eye trackers while introducing key eye-typing concepts to clarify essential terminology. We also examine related works by reviewing existing techniques to identify their strengths and limitations. Chapter 3 focuses on our virtual keyboard, explaining its design and functionality. Then, in Chapter 4, we describe our testing methods and analyse the results. Finally, Chapter 5 concludes the document by presenting our conclusion and future work.

Chapter 2

Background and Related Work

In this chapter we cover the different types of eye trackers and the terms necessary to understand why it can be a challenge to build a virtual keyboard. We also look into the related works to study the different ways to interact with the computer through gaze.

2.1 Eye Trackers

Eye trackers are devices that capture eye movements, usually with video cameras. Typically, they require a calibration process to estimate the users point of gaze on the screen. This gaze point is translated into X and Y coordinates, which the programs can then use to track and respond to the users visual focus.

Most modern eye tracking systems fall into one of four categories: Head-stabilized, remote, mobile (head-mounted), and embedded (integrated) [28].

2.1.1 Head-Stabilized Eye Tracking

These eye tracking systems often use methods to restrict head movements, typically with devices like bite bars or chin rests as seen in Figure 2.1. These high-precision systems are commonly used in fields such as neurophysiology or vision studies, where the focus on accuracy and precision takes precedence over participant comfort. This setup can achieve high accuracy by stabilizing the head, which simplifies tracking by reducing redundancy and noise in the data. Since the head is stabilized, a high resolution camera can capture a closer, detailed image of the eye without needing to adjust for head movements.

However, it can be uncomfortable, as it restricts the users head movement. Despite this drawback, it remains ideal for techniques that require rapid eye movements or the highest possible accuracy, though it is less suitable for our project, as our focus is on maximizing user comfort.

2.1.2 Remote Eye Tracking

.

Remote eye tracking systems operate without any direct contact with the participant. They work by positioning a camera from a distance, usually underneath the monitor, with a clear view

Figure 2.1: Head-Stabilized Eye Tracker (Figure Adapted From [28]).

of the eyes. They track the center of the pupil and the cornea reflection to determine both the eye position and head orientation, enabling free head movement while maintaining accurate gaze tracking.

As shown in Figure 2.2, these systems operate within a designated functional area, known as the "head box". If the user moves out of the head box or looks beyond the calibrated plane, tracking is briefly paused until the camera detects the user inside the box again.

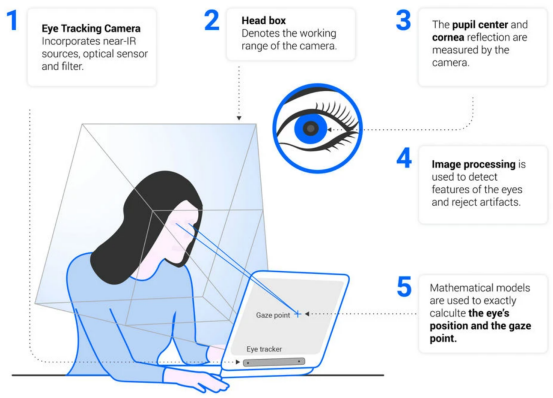


Figure 2.2: Remote Eye Tracking Detection Range (Figure Adapted From [28]).

This approach is well-suited to our project, allowing the user to interact with the computer comfortably and naturally while still accurately capturing gaze. The main downsides of this tracker include slightly lower accuracy compared to fixed systems and potential data gaps in data if there is excessive head movement. The latter can be minimized with the right implementation of the virtual keyboard.

2.1.3 Mobile Eye Tracking

This type of system is a head-mounted device worn by the participant. It can be in the form of glasses or headband, featuring a camera aligned with the visual path of the user to detect gaze direction, as seen in Figure 2.3. It is primarily used in real-world scenarios, such as sports or driving, where a comfortable, minimally invasive setup is essential for capturing natural gaze behavior.

Figure 2.3: Mobile Eye Tracking Device in Real-World Experiments (Figure Adapted From [28]).

While this system offers a high level of comfort and can operate freely without the constraints of a "head-box" as required by previous systems, it lacks a fixed coordinate system. Since it continuously gathers data regardless of the users gaze direction, it cannot determine whether the user is looking at the screen. Instead, it provides coordinates relative to the eyes position within the glasses. This setup records gaze data within a scene-based coordinate system defined by the scene camera, acting like an imaginary screen that shifts with the participants head. These reasons paired with the high costs associated with the device make it hard to implement in our project.

2.1.4 Integrated or Embedded Systems

Integrated systems come with built-in eye-tracking technology, like those found in virtual reality headsets seen in Figure 2.4. These systems offer high precision and are well-suited to their specific devices, providing intuitive control for menus within Augmented Reality and Virtual Reality environments where traditional input methods, such as a mouse or keyboard, are unavailable. However, their design specificity often limits their adaptability for wider or alternative applications.

Figure 2.4: Integrated Eye Tracker in Virtual Reality Headset (Figure Adapted From [28]).

This system is also unsuitable for our purposes due to the complexity of adapting it and the high cost associated with the product.

In conclusion, the most suitable eye-tracking device for our needs is a Remote Eye Tracker. It offers relatively good accuracy, straightforward setup, and a low-cost, making it more accessible

and allowing users the freedom of natural head movements, resulting in a more comfortable overall experience.

2.2 Selection Methods using Gaze

With the ability to accurately detect gaze and translate it into screen coordinates, we can effectively manipulate these coordinates to identify which keys the user intends to select for interacting with the computer. When considering how to select keys using gaze, the instinctive thought might be to blink. However, this method presents challenges since our natural blinking occurs frequently and unpredictably, making it difficult to distinguish between involuntary blinks and intentional selection blinks. Additionally, sustained blinking can lead to fatigue over time and an average person would be tired of blinking so much.

One of the more popular selection methods is Dwell Time, where users focus on a key without diverting their gaze for a predetermined period, typically ranging from 450 to 1000 milliseconds [37]. Once this time elapses, the key is selected. While this method is straightforward and has a low error rate, it does impose a limitation on typing speed due to the necessary waiting period for key selection.

To enhance efficiency, we can eliminate the dwell time requirement, leading to dwell-free interaction methods. However, this introduces a new challenge known as the "Midas touch"[16] which occurs when users unintentionally select a key simply by glancing at it. For instance, if users briefly look at the keyboard to orient themselves, they might inadvertently select a key with just a quick glance.

To mitigate this issue, researchers have explored using other techniques such as saccadic eye movements, gestures and pursuit.

Virtual keyboards based on the first method usually contain two sets of buttons, one which is the main letter key and another smaller button that shows up above the main button after gazing at the intended letter key. In order to select a letter a user would need to perform a saccade, which is a quick movement where they move their eyes up at the small button above the main button and then back down at the intended key to confirm selection. While this technique can still lead to unintended inputs if not executed correctly, it reduces the frequency of such occurrences compared to previous methods [40]. This makes selection fast, yet dizzy over a prolongated use due to the constant back and forth eye movement, and reduces the error rate given the extra steps it takes to select a letter [46]. If the eye tracker is of low quality and lacks precision, executing quick and accurate movements can be quite frustrating, as many of these actions may not be registered as intended. This can hinder the overall user experience and reduce the effectiveness of the typing method.

Gestures, on the other hand, allow users to select letters by performing specific eye movements, such as looking from left to right or shifting gaze from one box to another in a defined sequence. Unlike saccadic techniques, which require the user to move their eyes to a new location and then return to the original point, gestures, when used correctly, can eliminate this extra movement.

Gaze gestures are based on the path of the eyes rather than the exact fixation point, which helps reduce problems with tracking accuracy [20]. Unlike dwell-time methods, they also do not require the user to keep their eyes still for a set period. This method is great for comfort and does not cause much fatigue.

Another technique that does not rely on dwell time is pursuits. Similar to gestures, this method requires the user to perform specific eye movements. However, in this case, the movement is shown on the screen in the form of a moving object, and the user must follow the object, matching both pace and direction. This technique is called Smooth Pursuit, which is a eye movement that originates from the eyes following a moving object. This object is necessary because our eyes cannot replicate this movement without an external stimuli to follow [12]. Each letter is associated with a unique moving object, which is displayed visually. To select a letter, the user follows the corresponding motion, whether it is a circular movement to the right, a horizontal scan from left to right, or another pattern, ensuring that the speed and timing match the displayed movement. Since this technique relies on eye movements the accuracy of the eye tracker is not important. But, on the other hand the mental fatigue increases [3].

2.3 Performance Metrics

When comparing eye typing techniques and evaluating overall performance, researchers rely on established metrics to assess text entry efficiency and error rates. These metrics provide a standardized way to measure typing speed, accuracy, and user experience, ensuring consistent and meaningful comparisons across different methods and studies.

2.3.1 Text Entry Rate

To evaluate text entry rate, researchers commonly use Words Per Minute (WPM) as a standard metric [38]. This represents the number of words a user types within a given time frame. For consistency, a "word" is typically defined as any sequence of five characters, including spaces. This definition allows for fair comparisons across various techniques.

$$\text{WPM} = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5} \tag{2.1}$$

This metric provides insight into the speed of the typing process. The higher the WPM, the more efficient and faster the technique. To calculate WPM, the length of the final typed string minus 1 (because the time only starts counting when the first character is introduced so that character does not count) is divided by the total time taken (in seconds), then multiplied by 60 to convert it to a per-minute rate. Finally this value is multiplied by 1/5 to account for the standard five-character word length as seen in Equation 2.1.

We adapted the standard formula by omitting the "-1" term. This modification is necessary because our technique allows the users first input to be a complete word rather than a single character and users have to press a key to start writing.

$$\text{KSPC} = \frac{KeyStrokes}{|T|} \tag{2.2}$$

Another way to capture the efficiency of the typing process, is the amount of keystrokes needed to type the whole sentence. To address this, we use the Keystrokes Per Character (KSPC) metric [10], which measures the average number of keystrokes required to produce each character in the final transcribed text. As we can see in Equation 2.2 it is calculated by dividing the total keystrokes needed (counting backspaces and delete operations), by the length of the final transcribed string ($|T|$). Since our system incorporates word prediction, an ideal KSPC value is below 1, indicating that entire sentences can be typed with fewer keystrokes than the number of characters, demonstrating the efficiency of the application.

2.3.2 Error Rate

The Error Rate quantifies the proportion of errors corrected in the final typed string. This metric is calculated using the Minimum String Distance (MSD) [45], which determines the number of operations—deletion, insertion, or substitution—required to transform one string into another.

$$\text{MSD Error Rate} = \frac{MSD(P,T)}{MAX(|P|,|T|)} \tag{2.3}$$

The MSD provides a clear measure of accuracy by evaluating how much editing is needed to align the typed output (transcribed string) with the intended input (presented string). The formula for calculating the error rate via MSD is shown in Equation 2.3. In this formula, "P" represents the presented string, while "T" represents the transcribed string, and the denominator uses the length of the longer string. This normalization ensures the error rate does not exceed 1.00 while appropriately penalizing situations where the user types more text than was presented. This method allows for a fair and consistent evaluation of error correction across techniques.

2.4 Related Work

Currently the most used eye typing techniques can be put into two categories: Dwell-Based Eye Typing and Dwell Free Eye Typing [16].

2.4.1 Dwell-Based Eye Typing

The most common eye typing method is using a standard QWERTY keyboard where users select a key by gazing at it for a set amount of time. Once this time elapses, the key is selected. This process is slow, and the typing rate is bounded by the dwell time required for key selection.

Gaze-based text entry using dwell time tends to be slow, typically achieving speeds of about 5-10 words per minute (WPM). These findings are drawn from experiments involving beginners, using a fixed dwell time.

To improve the typing rate, Adjustable Dwell time [26] introduced a feature beneath the keyboard. It allows users to customize the waiting time before a key is selected by pressing plus or

minus buttons to decrease or increase the dwell time accordingly to their accommodations. This

Figure 2.5: Fast Gaze Typing Keyboard With Adjustable Dwell Time (Figure extracted from [26]).

method improved the standard typing rate from 5-10 words per minute to an average of 19.9 wpm in the final testing session, while reducing the dwell time from an initial average of 876 ms to 282 ms. This demonstrates that the method is an effective solution for increasing typing speed, while also being comfortable for users as it allows them to adjust the dwell time to their ideal preference.

In order to speed up the process, visual and auditory feedback has been used to enhance typing speed and accuracy in dwell-based eye typing systems [27]. Users receive feedback when selecting keys through an audio cue, similar to a clicking sound, while the key is visually highlighted (2nd key on Figure 2.6), and its symbol shrinks as the dwell time elapses. This helps users confirm their selections more efficiently.

Figure 2.6: Highlight Animation of Visual Feedback Mode (Figure extracted from [27]).

Other approaches attempt to dynamically adjust the dwell time [35] using an n-gram based letter predictor from n = 1 to n = 5 which updates continuously using Bayes rule. The uni-gram (n = 1) computes the probability of the first letter of each word and the bigram (n = 2) the second letter, and so on until 5. For words bigger than 5 they use the n = 5 model as well. This method gives us the predicted likelihood of the next letters given the past gaze and typing history, reducing the dwell time accordingly.

As we can see in the Figure 2.7 the keyboard follows a standard QWERTY layout, featuring a “send” key in the lower-left corner instead of the typical “enter” key, emphasizing that inputs cannot be modified once submitted. Additionally, the top panel provides system prompts and shows the users typed responses. Key selection provides both visual and audio feedback accelerating the process. An animated circle gradually closes around the chosen letter, indicating progress toward selection. Once the circle fully closes the key is selected, and a ’click’ sound is played.

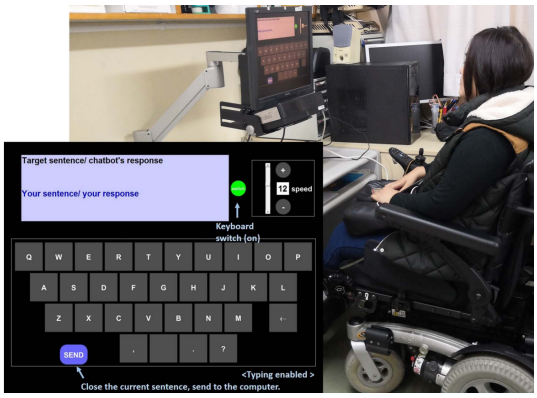


Figure 2.7: Dynamic Dwell Time Keyboard Layout (Figure extracted from [35]).

However the users felt tired sometimes and needed a place to rest their eyes without accidently triggering any buttons. To that end, a "switch" button, located on the right of the top panel, was added which allows users to enable or disable the keyboard. When the keyboard is disabled it pauses the task, allowing users to look around the screen worry-free. The interface also features a speedometer in the upper-right corner, allowing users to adjust the nominal dwell time. Increasing the number decreases the dwell time, while decreasing the number extends it, giving users control over the selection speed of the letters not affected by the dynamic dwell time. This technology had good typing rate results,with an overall 18.4 wpm and an improve of 26.1% in speed compared to its precedors.

Other techniques have been used as well, such as word prediction based on past input history and the most frequent words in the English language [25]. As the users type, a list of possible word suggestions is displayed, allowing them to select and autocomplete the word, which improves typing speed and reduces search time.

Following this, GazeTheKey [39] designed a technique where predicted words were displayed directly on the keys of a standard QWERTY keyboard. The predicted word associated with each key corresponded to a continuation of the currently typed sequence. As shown in Figure 2.8, after the user typed "ca", each letter displayed a potential word that could be formed by adding that letter. For instance, the letter "u" displayed "cause," suggesting how the word might complete if "u" were selected. There is also some more words showing up above the keyboard in case the users could not find the one they wanted. In order to select the corresponding words associated with the keys the users have to gaze at that specific key for a longer duration. However, this introduced issues, such as the inability to select the same letter consecutively.

To address this, a repeat button was added to the bottom left of the keyboard, allowing users to gaze at it and repeat the last selected letter. The results indicated that 54.4% of the words were completed by gazing on the keys and the study found an average text entry rate of 9.34 words per minute making word prediction a big improvement for eye typing techniques.

An example of saccadic eye-based input is Context Switching [15], which uses a KKBoard (Keyboard Keyboard), replicating a QWERTY layout across two separate regions—one above

Figure 2.8: GazeTheKey Keyboard Word Selection (Figure extracted from [39]).

and one below, with a central space that displays the typed text (see Figure 2.9). This design allows users to comfortably explore the entire keyboard without the risk of unintentional selections ("Midas touch" problem).

In the Context Switching paradigm, two distinct eye movements are employed: a short fixation on a letter activates key-focus, while key selection (i.e., typing the letter) is performed by saccading from one context to the other. This separation between focusing on and selecting keys improves accuracy and reduces unintended inputs.

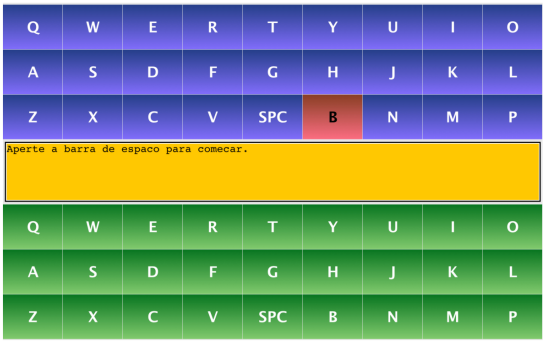


Figure 2.9: ContextSwitching KKboard Layout (Figure extracted from [15]).

To type using the KKBoard, the user first fixates on a letter in the upper keyboard for 150ms. Once the letter turns red, they must gaze at the same letter in the lower keyboard within 450ms to select it. The experiments demonstrate that KKBoard is intuitive and easy to learn. Users familiar with the QWERTY layout can quickly achieve typing speeds of over 10 words per minute, even without character prediction, making it a good beginner-friendly approach.

Some techniques deviate from the familiar QWERTY layout, requiring users to spend more time learning how to gaze-type effectively, which can also lead to increased fatigue due to the high levels of concentration involved. However, after a period of adjustment, many of these alternative keyboards prove to be beneficial, offering more comfort and even adding an element of enjoyment. These systems can employ dwell-time methods, dwell-free approaches, or rely on saccadic eye movements for input.

An example of these virtual keyboards is WordPop[10], which utilizes both saccading move-

ments and dwell time, depending on the users preference, offering a dwell time and dwell time free approach. As we can see in Figure 2.10 WordPop presents users with a set of six buttons positioned at the bottom of the interface. Each button represents a group of five letters arranged alphabetically from left to right, with one button dedicated to punctuation marks.

In order to select a letter, the user has to gaze at the button containing the desired letter and gaze at it for 250ms (dwell time) or reverse crossing into a smaller button positioned above the main button (saccade). To provide feedback, the selected button is highlighted in green, ensuring the user is aware that their input has been successfully registered.

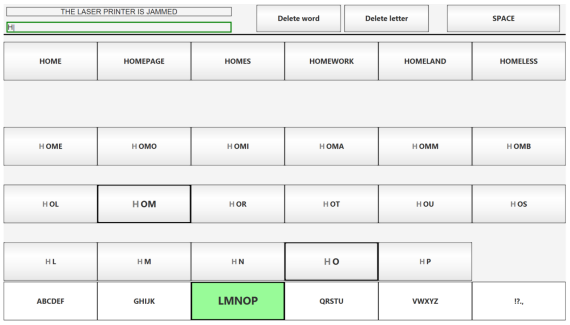


Figure 2.10: WordPop Keyboard Layout (Figure extracted from [10]).

After selecting a button, a new set of buttons appears directly above the initial row. Each button now represents one of the letters within the selected group. For example, in Figure 2.10, we see a depiction of the word “Home” being typed. The user had already selected “H” from the first row and is now choosing the second letter. He has already chosen the selected group he wants, thus the green highlight, and a second row displaying all the letters from the group appeared above. In this case the user selected “O” with either of the selection methods mentioned before, and a darker box highlights around the button to show it was selected or is being currently focused. After choosing the letter the system introduces another row above the current one, consisting of predicted letters that connect with the chosen one to make valid English word. This prediction takes into account all the past letters after the last space, such as the “H” in this example. Afterwards the system provided the user with the predicted letters row above in which he selected “M”. These letters are predicted by the most frequent use in English Language, with the most frequent appearing to the left. This way the user builds his word in a staircase, always climbing.

At the top of the staircase there is a row displaying fully predicted words, which the user can select instead of typing it and speed up the process. These slots are filled with suggested words sorted and calculated the same way the letters were. These words can be selected by gazing at them with a Dwell Time, and a space will be automatically placed after the word to have a better quality of life and increase speed. Above this row, at the very top of the interface, there are two lines. The first line displays the phrase the user is meant to type, providing a reference for accuracy. The second line shows the text the user has already typed, allowing them to track their progress in real time. Additionally, three buttons are located on the top right corner of the interface, offering essential functionalities. Them being a “Delete Word” button to allow the user to delete an entire

word in a single action, a “Delete Letter” which allows the user to delete the last typed letter, and “Space” to enter a space. These three buttons were put on the top row to be easily accessible.

After testing this technique, researchers found that participants achieved faster typing speeds using the Dwell-Time selection method compared to the Reverse Crossing method. On average, participants typed at a rate of 5.5 words per minute (WPM) with Dwell Time, while the Reverse Crossing method achieved an average of 4.86 WPM. Dwell Time was on top in terms of user preference and comfort, suggesting that the approach may provide a smoother and more comfortable typing experience for users since Reverse Crossing requires more eye movement and thus is more tiring on the eyes. However Reverse Crossing achieved a lower error rate of 0.19% compared to the 0.57% error rate of Dwell Time.

Some users may have disabilities that impair smooth eye movement, making the standard QWERTY keyboard ineffective for them. An alternative could be a one-dimensional keyboard [9] allowing users to easily type using only their gaze, even with erratic movements. The keyboard functions by displaying a vertical column containing all the letters of the alphabet, with a three-letter selector (see Figure 2.11. Users can scroll up and down through the letters, and when they reach the desired letter, they gaze at it. A green outline provides visual feedback on the selection. Since the selector highlights three letters at a time, the user technically selects all three, and the system then attempts to infer the intended word based on the chosen letters. A list of previously selected letters appears to the left of the keyboard and a list of predicted words appears to the right of the keyboard, allowing the user to later select the correct word if it is included in the suggestions by gazing into the box at the bottom.

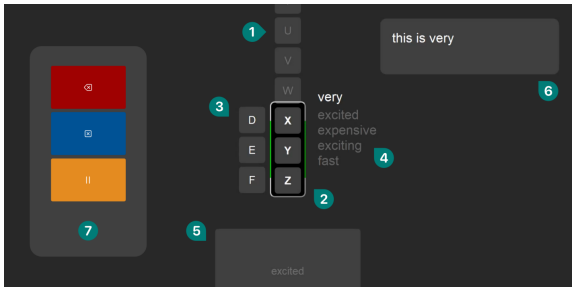


Figure 2.11: One-Dimensional Keyboard (Figure extracted from [9]).

This keyboard displays a delete, clear and pause button to the left. Participants achieved 3.75 WPM without prediction, and 11.36 WPM with prediction, gaining a 202.9% increase.

Another technique worth mentioning is the Adaptive dwell time for eye typing [19] which focuses on improving the efficiency of gaze-based key selection with adaptive dwell time. This technique adjusts the selection time based on the users typing efficiency. The system continuously monitors typing behaviour, adapting the dwell time parameter based on error rates and input speed. If the number of errors made, divided by the number of all typing actions, is greater than a given threshold, the time of gaze may be increased. Conversely, if the average time between two consecutive commands is close to the current dwell time, typing is considered efficient, and the time can be reduced. To evaluate this approach, the authors tested three existing typing methods,

each represented by a different virtual keyboard implementation. They are similar in entering text procedure but differ in the key layout and realization of some actions. The first virtual keyboard developed by Huckauf and Urbina [42] can be seen in Figure 2.12. This design groups letters into the pie items. At first, only one pie with letter groups is visible on the screen. When the user gazes at a group, a secondary pie appears, with each sector representing a single character from that group allowing users to select a letter. A textbox on the right side of the keyboard displays the entered text, while the center of the pie holds the Back button. Additionally, a Delete button for error correction is located in the bottom pie item.

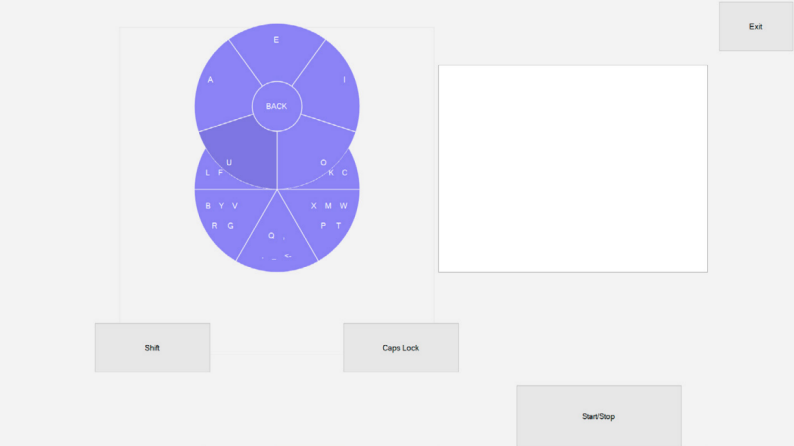


Figure 2.12: Pie Keyboard (Figure extracted from [19]).

The second virtual keyboard, developed by Cecotti, Meena, and Prasad [6], is shown in Figure 2.13. This design places rectangular buttons around the edges of the screen. Initially, these buttons display groups of letters in both uppercase and lowercase. Once a group is selected, the buttons update to display the individual characters within that group, enabling precise letter selection. The letters are arranged alphabetically for ease of navigation. In this layout, the Delete button changes its function to “Back” when the user is in the letter-selection state, allowing them to undo a choice and return to the group-selection phase. A textbox displaying the entered text is positioned at the center of the screen.

Figure 2.13: Tree Keyboard (Figure extracted from [19]).

The final approach, developed by Harezlak, Basek, and Kasprowski, is called Side [18]. This virtual keyboard, illustrated in Figure 2.14, was created after studying the other two designs. It resembles the second keyboard in that it includes a central text box, but differs in its layout: instead of placing buttons around the screen, Side positions them on the left and right edges. The buttons are triangle-shaped due to the optimization of the space to fit as many groups of letters as possible in the smallest possible plane while, at the same time, maintaining a large area for easy viewing.

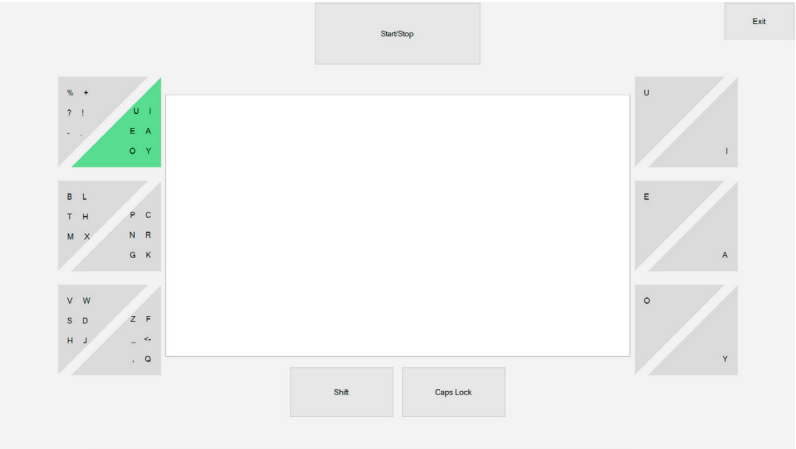
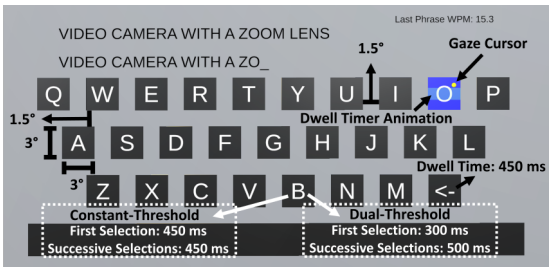


Figure 2.14: Side Keyboard (Figure extracted from [19]).

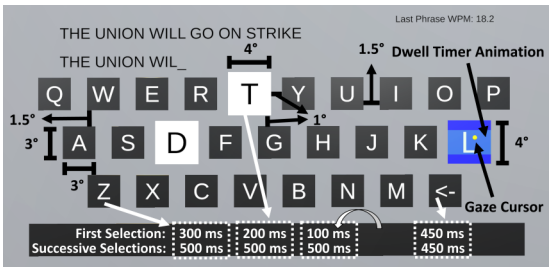
Text entry is performed in two steps. First, the user selects a group of characters from the left side of the screen, which opens a corresponding panel on the right containing the individual letters on separate buttons. In the second step, the user selects the desired letter. Once selected, the character is entered into the text field, and the right-side panel is hidden to provide clear visual feedback that the entry has been successfully made.

The authors conducted a study with seven participants to evaluate the virtual keyboards under different conditions: a fixed dwell time of two seconds, a fixed dwell time of one second, and the adaptive dwell time. The results showed that, when averaging across all keyboards, the two-second dwell time achieved a typing speed of 1.63 WPM, the one-second dwell time reached 2.38 WPM, and the adaptive dwell time produced 1.93 WPM. In terms of errors, however, the adaptive dwell time proved more reliable, with a total of 173 errors across all keyboards, compared to 280 errors for the two-second dwell time and 533 errors for the one-second dwell time. Showing that shorter gaze durations can have fewer errors at the cost of speed, achieving a balance between extremes.

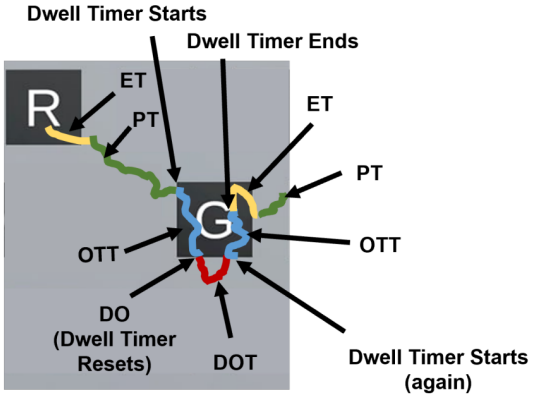
Recently, a study on Multi-Threshold Dwell Times was conducted [29], in which a virtual keyboard supporting three types of dwell times was developed (Figure 2.15): a standard Constant-Threshold Dwell time (CTD), a Dual-Threshold Dwell time (DTD), and a Multi-Threshold Dwell time (MTD). The research aimed to investigate why dwell-time text entry methods struggle to surpass 20 WPM and to identify which part of the dwelling process causes the greatest time loss.



(a) Constant and Dual Threshold Dwell keyboards (Figure extracted from [29]).



(b) Multi-Threshold Dwell (Figure extracted from [29]).



(c) The Components of Dwell Selection (Figure extracted from [29]).

Figure 2.15: Multi-Threshold Dwell Technique: (a, b) Keyboards design (c) Dwell Selection.

In Figure 2.15a, we can see the layout of the virtual keyboard. At the top, the sentence to be typed and the text already written are displayed. Below that is the keyboard area, where each key has a width and height of 3°, with a gap of 1.5° between adjacent keys to prevent unintended selections caused by low tracking accuracy. When the users gaze makes contact with a key, the system highlights that key in blue and starts making an animation indicating the dwell time progress. Once the dwell threshold is reached, the key is selected, and the system provides both visual and auditory confirmation, highlighting the selected key in green for 100 ms.

To determine which parts of the dwelling process consume the most time, an initial study was conducted using a constant dwell time of 450 ms with nine participants for eight days. To analyze this, multiple metrics beyond the usual text entry rate metrics where used and can be seen in Figure 2.15c, these metrics represent:

Exit Time (ET): The time taken to leave a key after it has been selected. **Pointing Time (PT):** The time required to move the gaze from the previously selected key to the next one.

Activation Time(AT): The time taken to activate a key, calculated as the sum of two components: On Target Time (OTT) and Drop-Off Time (DOT). **On Target Time (OTT):** The time spent fixating on a key (dwell time). Jitter is inevitable in eye tracking and is the main reason for unintended resets of the 450 ms dwell timer. **Drop-Off Time (DOT):** The time lost due to gaze jitter which is when the gaze involuntarily drifts off the selected key after first landing on it. That

means the total time (TT) for selection is the sum of all these components.

The results from the first study showed that the average text entry rate increased from 11.6 WPM to 14.8 WPM. The Pointing Time (PT) and Exit Time (ET) were the components of the dwell interaction that demonstrated the greatest improvement in reducing selection time. This suggests that participants became more familiar with the exact key locations and improved their ability to anticipate when the dwell threshold would be reached, allowing them to move toward the next key more efficiently with practice.

Guided by the results of the first study, the authors designed two new dwell-based keyboards: one implementing Dual-Threshold Dwell and another using Multi-Threshold Dwell. The Dual-Threshold Dwell keyboard, shown in Figure 2.15a, is similar to the original version, however, it has two dwell durations, a 300 ms dwell time for the initial selection, followed by an increased 500 ms dwell time for the same key. This adjustment helps prevent accidental repeated selections of the same character caused by shorter dwell times. The Multi-Threshold keyboard shown in Figure 2.15b incorporates two fixed dwell thresholds of 300 ms and 200 ms.

The first letter of any word must be selected by dwelling on the key for 300 ms. After this selection, the three most probable next letters are highlighted and enlarged, and only these highlighted letters can be selected using a 200 ms dwell time. This method retains the same design principles as the previous keyboards, meaning that consecutive identical letters require a 500 ms dwell time, while the Backspace key uses 450 ms. Since the Space bar is the most frequently used key, its dwell time was reduced to 100 ms but increased to 500 ms for consecutive selections.

To evaluate these changes, the authors conducted a second study with 15 participants, testing both the Dual-Threshold and Multi-Threshold keyboards. The results showed an average typing speed of 18.3 WPM for the Multi-Threshold keyboard and 15.3 WPM for the Dual-Threshold keyboard, with mean dwell time thresholds of 255 ms and 356 ms, respectively. These findings demonstrate that shorter dwell times can significantly enhance typing speed when applied carefully. The authors also administered a feedback questionnaire, which revealed that participants found the Multi-Threshold keyboard more frustrating and fatiguing, mainly due to its increased complexity and higher attention demands. Additionally, the Pointing Time (PT) and Exit Time (ET) values were comparable across keyboards, indicating that the observed speed improvements primarily came from the reduced dwell times rather than faster transitions between keys.

2.4.2 Dwell Free Eye Typing

Dwell-Free Eye Typing systems eliminate the need for dwell time to select keys, which can introduce several issues during selection process. According to GazeTry [23] these errors can be characterized as extra-letter errors, neighbour-letter errors, and missing-letter errors. The extra-letter error, also known as the “Midas touch”, occurs when unintended keys are selected due to the absence or reduction of dwell time (Ex.: Eye Tyyping). Neighbour-letter errors happen when users accidentally gaze at an adjacent letter instead of the intended one (Ex.: Eye Typong), while

missing-letter errors arise when users fail to fixate on the correct letter (Ex.: Eye Tping).

To address the removal of dwell time, many systems allow users to type by gazing sequentially at the letters of a word. GazeTry does this by selecting the "dominant letters"—those on which users spend more time during the swipe entry process. If the word does not match any existing words, the input string is passed through a "String Matching Module", which finds the most related string by calculating the cost of converting one string into another. Unfortunately the authors do not present typing rates results from their tests, leaving only an idea of the techniques potential.

Filteryedping goes a step further [34]. It consists of a QWERTY keyboard where the users type by gazing sequentially at the letters but with an algorithm that generates a list of possible words the user may have intended to type, allowing them to select the correct word from the list. (see Figure 2.16). The user begins by gazing at the letters of the word one by one, receiving visual feedback for each selected letter as well as auditory feedback that reads the typed word aloud. Once the input is complete, the system displays the recognized letters and suggests a list of words underneath the keyboard that closely match the intended word. If the desired word is not listed, the user can press an arrow in the bottom right corner to reveal additional options. The upward arrow in the top left represents the shift key for uppercase letters, while the side arrow in the top right functions as the delete button. Once the correct word appears in the list, the user gazes at it, confirming their choice by gazing back at the keyboard meaning they are satisfied with the selected word and thus finalizing their selection. Tests with participants showed this method achieved an average typing speed of 15.95 words per minute, making it faster than some previous techniques.

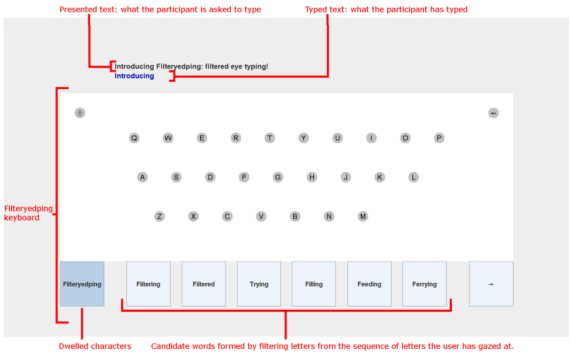


Figure 2.16: Filteryedping Keyboard Layout (Figure extracted from [34]).

In 2021, a new technique named Head-Gesture Assisted Gaze Typing (HGaze) [14] was introduced, combining both gaze-based and head-based input that provides a more accurate and stable typing. This method divides tasks into three categories: (1) head gestures for tasks requiring high precision, (2) head gestures for commands related to visual activities, and (3) gaze input for tasks that can be completed with quick movements and lower accuracy demands. For example, to delete words, users rotate their heads left or right, and to navigate the word list, they tilt their heads left or right, while the gaze is used in the word typing process. The HGaze interface includes a text box, a cancel/delete indicator, a virtual keyboard, and a confirmation key as seen in Figure 2.17. To enter a word, the user selects the first letter by fixating on it and nodding, then a red dot will

appear providing visual feedback of the fixation.

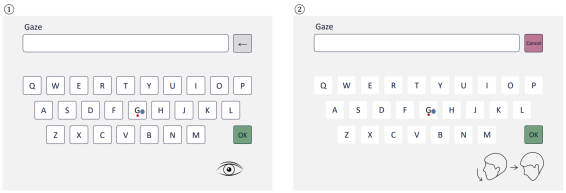


Figure 2.17: HGaze Demonstration: Steps 1-2 (Figure extracted from [14]).

The users then glance over the rest of the letters that form the word until reaching the last letter, in which they will nod again to confirm. Another red dot is shown, and after the last letter is selected a list of five word candidates appears above it, with the most probable word highlighted and automatically entered in the text box as seen in Fig. 2.18.

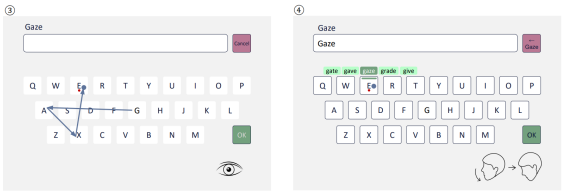


Figure 2.18: HGaze Demonstration: Steps 3-4 (Figure extracted from [14]).

In case it was not the correct word the user can tilt their head left or right to scroll through the word options and select an alternative if needed. If the desired word is not in the candidate list, the user can delete the typed word by shaking their head. This same gesture also allows the user to cancel the current word path if a mistake was made. A cancel/delete indicator is positioned to the right of the text box, providing a visual reference for the user to confirm the cancel/delete status, giving intuitive feedback and helping users maintain control over their input to avoid mistakes.

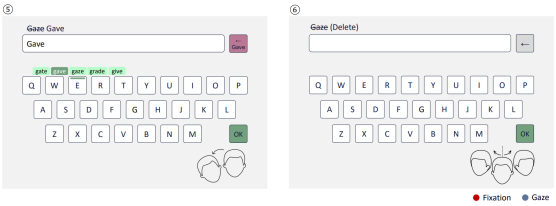


Figure 2.19: HGaze Demonstration: Steps 5-6 (Figure extracted from [14]).

After an extensive testing session, the authors achieved an overall typing rate of 11.22 words per minute with the HGaze system, compared to 9.53 wpm using a traditional dwell-time keyboard with the same participants, and a below 3% MSD error rate. One of the downsides of these techniques is that they do not work for words that are not in the dictionary since most of them derives from a list of possible words, so they must be input by reverting to dwell-based techniques.

Since pursuits do not require waiting time they also make a good candidate for dwell free eye typing. This was latter tested and compared in a study with three other gaze interaction methods, Dwell Time, Pursuits, and Gaze gestures, on mobile devices [30].

This study evaluates these methods and the goal was to measure both performance and user perception of these techniques when used on handheld devices. A second goal was to examine how these methods perform while participants are walking, as a step toward enabling on-the-go gaze interaction. The interface used for testing was a virtual keyboard, where letters are arranged in rows and columns in alphabetical order, as shown in Figure 2.20.

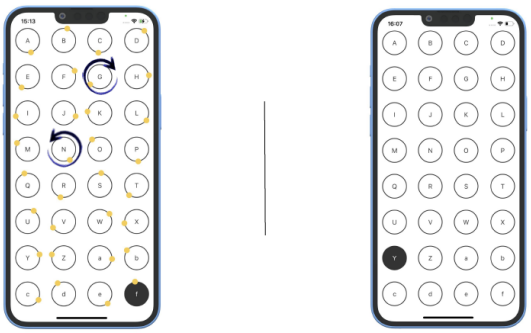


Figure 2.20: Virtual Keyboard for Pursuit Selection (left) and Dwell Time selection (right) (Figure extracted from [30]).

Each of the techniques has a different selection method but the keyboard layout remains the same in all of them.

Dwell-time selection works by fixating on a target for a set duration. To make a selection, the user gazes at the desired object and maintains fixation for at least 800 ms. The system then confirms the selection by calculating the mean fixation point within this time window (Figure 2.20).

This method requires very accurate gaze estimation and dependent on good calibration. On handheld devices, frequent movement makes it likely that calibration will drift, reducing reliability.

In the Pursuits method, a small object revolves around each letter. Selection is made by aligning the eyes movement with the trajectory of the target object. To provide input, the user (1) identifies the desired letter and (2) briefly follows the orbiting object with their eyes. The system then determines the selected target by measuring how closely the users eye movements match the objects path. This similarity is calculated using the Pearson correlation coefficient (Figure 2.20).

An advantage of this method is that it does not require accurate gaze estimates or calibration because it relies on smooth pursuit eye movements to follow moving objects. However, it can introduce greater visual and cognitive load, as users must track moving stimuli continuously, which may lead to fatigue during longer interactions[3].

Gaze gestures are eye movements that follow specific patterns in a sequential time order to issue commands or perform selections. Their implementation of gaze gestures follows the approach of Look to Speak by Google [13]. To provide input, the user (1) locates the target on the screen, either on the left or right side, (2) performs a gesture (right or left) to select the side containing the desired target, and (3) repeats this process until the desired target is reached. Gesture selection is recognized when a single right or left gesture is performed within a 1000 ms (30 samples) time

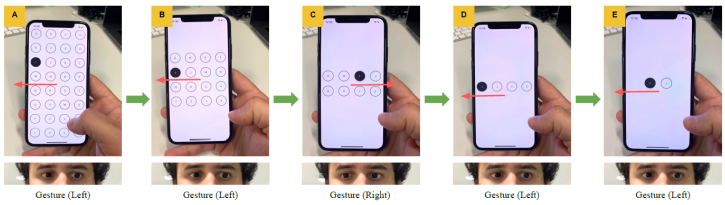


Figure 2.21: Virtual Keyboard for Gestures selection (Figure extracted from [30]).

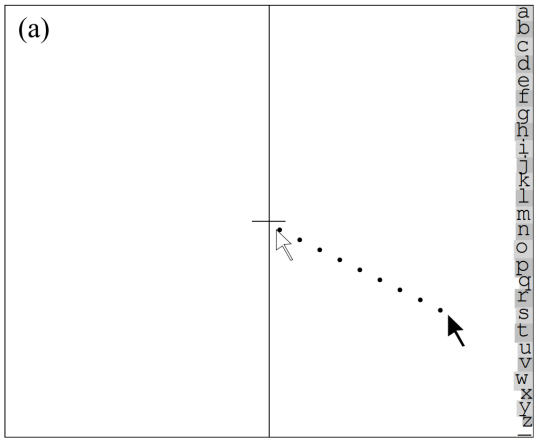
window. This process is illustrated in Figure 2.21. The user first selects the side of the keyboard containing the desired letter. The keyboard then narrows down step by step until only two letters remain. A final left or right gesture selects the intended letter.

This technique does not rely on precise gaze or frequent calibration, since it is based on directional gestures. A key advantage of gaze gestures is their independence from the exact screen state, and by extending the gesture set, the system can support a larger range of commands.

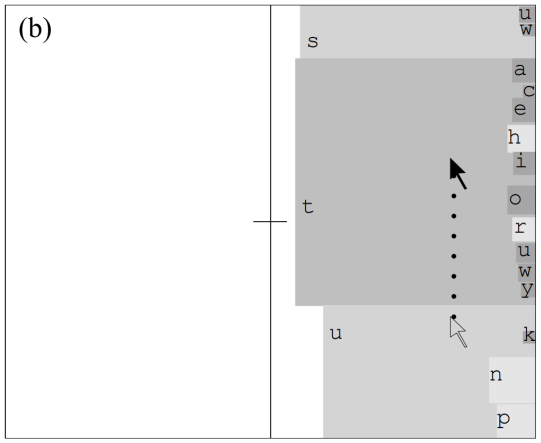
To understand which of the techniques was better they made a study with 24 participants. Participants performed selection tasks while sitting and walking, selecting from 2, 4, 9, 12, and 32 on-screen targets. The study measured selection time, error counts, timeouts, perceived cognitive load, and user preference. The results revealed clear differences among the techniques. Pursuits had faster input than both Dwell time and Gaze gestures. Although Gaze gestures were the slowest and most demanding method, they achieved higher accuracy than both alternatives in both sitting and walking conditions. Participant feedback further showed that Pursuits imposed lower mental and physical demand, requiring less effort and causing less frustration compared to the other methods. In terms of user preference, Pursuits were favored when stationary, while Dwell time was preferred when walking. Compared to traditional eye tracking techniques designed for stationary use, these findings highlight the trade-offs that emerge in mobile contexts. While Pursuits provide faster input and lower workload, and Gaze gestures achieve higher accuracy, user preferences shift depending on mobility. Unlike conventional sitting scenarios where accuracy and stability dominate, handheld use introduces calibration challenges and movement effects that make Pursuits and Dwell time more practical choices for low accuracy scenarios.

Dwell-time–free approaches enable a wide range of unconventional keyboard designs, as they rely on alternative selection methods such as pursuit, swipe, gaze, or gestures. To support these methods, specialized keyboards are developed that are specifically tailored to each technique.

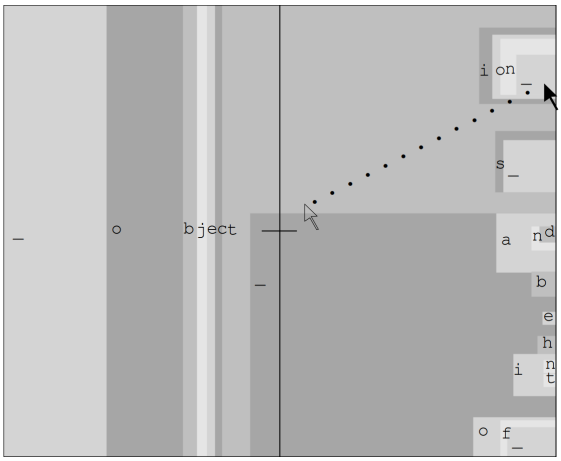
Dasher [22] is an example of an unconventional keyboard where letter selection is represented by dynamically resizing boxes, whose sizes changes based on the probability they will be selected. As seen in Figure 2.22, the interface starts by presenting a large box containing a column of 27 smaller boxes (Figure 2.22a), each representing a letter of the alphabet, plus the space represented by "_". The typing process begins by gesturing towards the desired letters rectangle. For example, for the word "the", the user moves towards the rectangle containing "t", which gradually enlarges, making it easier to select.



(a) Dasher's initial configuration (Figure extracted from [22]).



(b) Gesturing Towards "h" in a Dasher Keyboard (Figure extracted from [22]).



(c) Selecting Multiple Characters in Dasher (Figure extracted from [22]).

Figure 2.22: Dasher Technique: (a) Initial configuration, (b) Selection (c) Multiple Selection (Word Prediction).

Inside the newly enlarged box, smaller rectangles appear, representing the next possible letters to form a valid word. For example, after selecting "t", rectangles corresponding to combinations like "ta", "te", and "th" are displayed. These are organized so that the most frequently used and likely letters appear first and have larger boxes. Since "the" is a common and highly probable word in the English language, moving towards the "h" is straightforward, as it will likely have a larger rectangle, making the selection process faster and more intuitive.

Finally, with another simple gesture, the user can select "e" and complete the word. In Dasher, it is possible to select multiple letters with a single gesture, allowing users to type at higher speeds and increase typing rate. Dasher achieves this by calculating the most likely word completions based on a language model. For instance, as shown in Figure 2.22c, the string "object" has possible completions such as "objection", "objects_", "object_and", "object_of" and by gesturing towards "ion" the users completes the word. By leveraging the language model, Dasher helps users

complete their words more efficiently, reducing the likelihood of spelling errors.

Dasher's operation is intuitive and immediately clear to new users. Additionally, it has a steep learning curve, making it comparable to other text entry methods in terms of ease of use. In one of the sessions this technique achieved a top speed of nearly 150 characters per minute, which, considering an average word length of five letters, translates to an impressive 30 words per minute (wpm) making it a solid unconventional technique for faster typing.

Saccadic systems have also had a significant impact on gaze typing as an efficient selection method. One example is pEYE write, a saccadic eye-gazing technique [21]. In this system, the user starts by gazing at a pie menu divided into six groups, each containing a set of letters, with one group dedicated to special characters. The user selects the group that contains the desired letter, which opens a second pie chart displaying all the letters in that group, as shown in Figure 2.23.

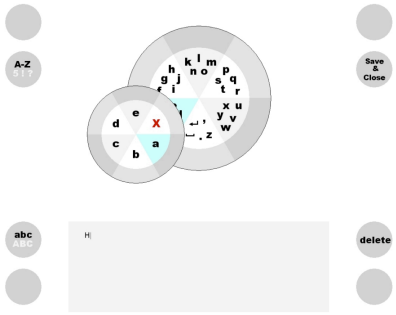


Figure 2.23: pEYE write Pie Menu (Figure extracted from [21]).

The user then gazes at the piece of the pie containing the desired letter and selects it by saccading outward to the frame of the pie and back inward. Additionally, the interface includes other functional buttons outside the pie menu, which the user can access using the same gaze method. By gazing into one of those buttons, then at a gray button that will appear, and back to the original button, the user can trigger actions like shift, delete, or save. This technique offers more versatile operations for gaze typing. The authors made a study with 4 participants over 5 sessions and achieved a good average of 12.33 WPM, although the test sample is too small to reasure of its results but still one of the fastest saccade techniques.

Pursuit-based techniques are most commonly applied on mobile devices [2][30], as they do not require calibration or rely heavily on eye-tracker accuracy. However, they can also be adapted for desktop applications, eliminating the need for calibration altogether. SPEye [36] and Leyenes [11] are both virtual keyboards that combine eye tracking with smooth pursuit to enable users to select input elements.

SPEye [36] is a calibration-free gaze writing technique based on smooth pursuit. Unlike the others this one does not require a single initial one-point calibration [48][33], making it fully working without the need to calibrate eye tracker.

(a) SPEye initial screen (dark mode) (Figure extracted from [36]).

(b) SPEye subscreen for the "ABCDE" group (light mode) (Figure extracted from [36]).

Figure 2.24: SPEYE Technique: (a) Layout and (b) Subscreen.

The interface of SPEye is shown in Figure 2.24a. It consists of a small central circle with radial beams extending outward, each ending with a label that represents a character, action, or group of characters. On the main screen, there are ten beams and labels: five for alphabet letters, one for numbers, two for the delete and space functions, one for symbols, and one for word predictions. This virtual keyboard is designed in two versions, one for dark mode and another for light mode allowing users to choose the option that is most comfortable for them and help reduce eye strain.

To select an element, the user must follow the moving circle along the corresponding beam using their gaze. To activate the motion, the user must first look at the center of the cluster for 500 ms. Once this is done, all circles begin moving outward along their respective lines. As the users gaze tracks the desired circle, both the circle and its associated label turn red, confirming selection. The user is then taken to a sub-screen containing the relevant letters, symbols, or numbers, as illustrated in Figure 2.24b.

By repeating the selection process, the user can choose individual letters to form words. To improve typing speed, the system also includes a word prediction feature that suggests possible words based on the letters already entered. As letters are selected, the system updates and displays a list of predicted words on the left side of the screen, as shown in Figure 2.25. If the intended word appears in the list, the user can select the line labeled "WORDS" to open a sub-screen containing all the predicted words for final selection.

Figure 2.25: SPEye initial screen with "Hel" typed (left) and subscreen for the word prediction group (right) (Figure extracted from [36]).

To evaluate the typing speed of SPEye, the authors conducted user tests with three volunteers over 22 sessions. Although the small sample size limits the generalizability of the results, the participants achieved average typing speeds between 0.90 and 1.15 WPM without using word prediction. Despite the speed, the system demonstrated a low error rate with "only one typing error" recorded [36]. The study concluded that SPEye is a suitable assistive technology for hands-free text entry, particularly in situations where maintaining accurate calibration over time is difficult, as it operates without requiring calibration. Additionally, the results showed that while SPEye is slower than other smooth pursuit-based writing systems, it is also more robust and reliable.

After SPEye[36] a similar technique named Leyenes[11] was created but instead of using radial lines they used horizontal lines. The interface is organized into horizontal lines, each also containing a small moving circle. The speed and direction of the circle differ across lines, enabling the selection algorithm to detect smooth pursuit movements and match the users gaze with the motion of a specific circle. Each line is labelled with the associated character, action, or group of characters, as shown in Figure 2.26. The interface also provides a delete key to remove the last typed character and a back key to return to the previous screen. At the top of the keyboard, a black bar displays the text being entered.

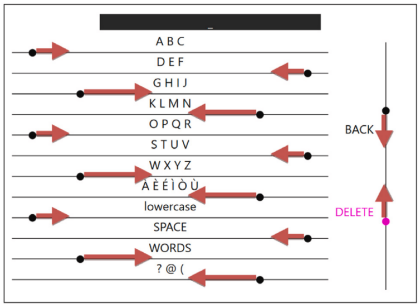


Figure 2.26: Leyenes Letter Screen Layout (Figure extracted from [11]).

To select an element, the user follows the moving circle on the corresponding line with their gaze, much like SPEye[36]. When the system detects a match between the users gaze direction, speed and the circles movement, the circles color changes to red. If the user maintains their gaze on the circle for a predefined time, the color changes to green, and the associated action is performed.

The application is structured into five main screens: $Initial$, $Home$, $Letters$, $Numbers$, and $Symbols$. The $Initial$ and $Home$ screen are used to start the application. The $Numbers$ screen (Figure 2.27, left), contains eleven lines representing digits, along with an additional line that provides access to the $Symbols$ screen (Figure 2.27, right).

The $Letters$ screen, is organized into twelve horizontal lines: eight dedicated to groups of letters, two for upper/lower case and space, one for symbols and punctuation, and one for word predictions (Figure 2.26). Choosing a group of letters opens a corresponding sub-screen containing lines for each letter in that group. By following the moving circle on the line of the desired letter, the user can select it, thereby typing it.

As mentioned earlier, this technique incorporates word prediction. On the $Letters$ screen, the

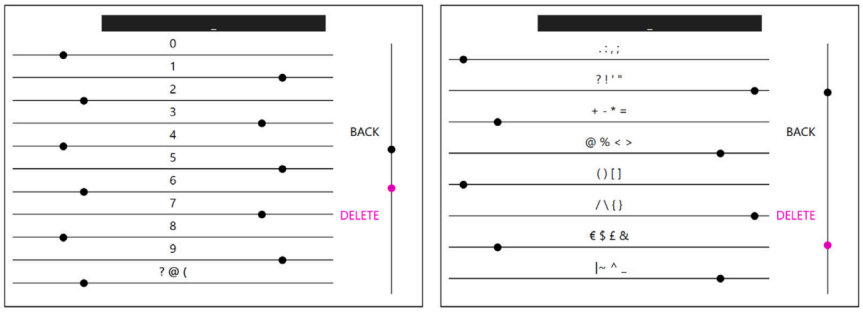
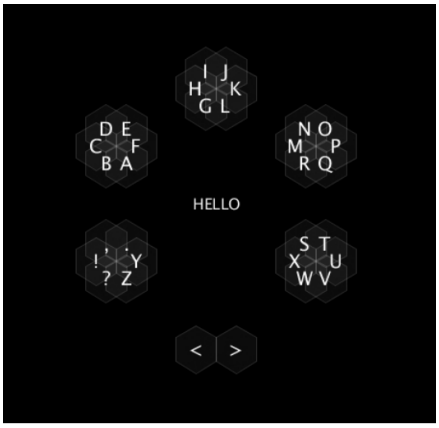


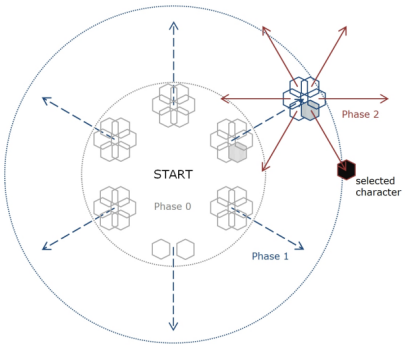
Figure 2.27: Leyenes Number Page (left) and Symbol Page (right)(Figure extracted from [11]).

four most probable words are displayed on a line with spaces between them. These predictions are generated based on the letters already entered.

A user study was conducted with 24 participants aged between 25 and 61, resulting in an average text entry rate of 1.32 WPM, being a little faster than SPEye [36]. The study highlighted several limitations. Leyenes proved to be a relatively slow text entry method and required a considerable amount of screen space to operate. However, it achieved a lower error rate compared to other smooth pursuit writing approaches, despite its slower speed.



(a) SMOOVS Layout (Figure extracted from [33]).



(b) SMOOVS Phases 0 - 2 (Figure extracted from [33]).

Figure 2.28: SMOOVS Technique: (a) Layout and (b) Phases.

Another pursuit-based technique worth mentioning is SMOOVS [33]. It is a gaze-based text speller that uses smooth pursuit eye movements to let users enter text. The system only needs a one-point calibration at the center of the screen and does not require much training to use.

The interface is made up of character elements arranged in a hexagonal layout with six clusters of characters. Each cluster contains up to six tiles, representing neighboring letters of the alphabet. The system begins in an idle phase, during which the clusters remain stationary, and the typed word is shown in the center area (Figure 2.28a).

The SMOOVS technique uses smooth pursuit eye movements to let users input text with minimal calibration, following four main phases. In Phase 0, the character tiles remain stationary while

the system waits for the user to either look at the central area (inactive state) or focus on one of the character clusters (active state). In Phase 1, the selected character clusters move outward from the center, and the system tracks the users gaze path to determine which cluster is being selected.

In Phase 2, if a valid pursuit movement is detected during Phase 1, the individual character tiles within the selected cluster move away from each other. A character is selected when the users gaze path matches the movement of a specific character tile. Meanwhile, the tiles from other clusters move slightly as shadows to maintain a dynamic visual effect.

Finally, in the last phase, all tiles return to their initial positions. Visual and auditory feedback are provided for the selected character in the form of high saturation and broader edges of the tile, and a short, soft sound is played. After this, the system returns to Phase 0 (inactive condition).

The test results showed that with increasing object movement speed, the number of completed gaze paths and the number of corrections also increased. However, at 340 px/s, the drawbacks of corrections outweighed the benefits of the higher number of gaze paths, thus the text entry rate could not be boosted by further increasing object movement speed. The highest text entry rate achieved in the experiment was 3.34 WPM at 300 px/s, indicating that moving objects too quickly on the screen can reduce overall efficiency.

2.5 Analyses and Discussion

Table 2.1 presents a summary of all the approaches studied. We can see that Dwell Time-based approaches tend to have lower error rates due to their accuracy, as they require users to fixate on each key for a specific duration, reducing unintentional selections. For instance, Adjustable Dwell Time [26] shows an average error rate of 0.82%, decreasing to 0.36% by the final sessions, while Dynamic Dwell Time [35] maintains an error rate below 1%. Despite their accuracy, both techniques achieve commendable typing speeds: Adjustable Dwell Time averages 15.7 WPM, with a notable increase from 6.9 WPM at the start to 19.9 WPM in later sessions. Dynamic Dwell Time, meanwhile, starts at an impressive 21.3 WPM but decreases to 16.6 WPM in the final sessions, as later tests introduced more complex and uncommon words. Adaptive Dwell Time [19] follows a similar principle but adjusts the fixation duration dynamically based on user performance, achieving an average of 1.93 WPM, which is considerably lower than other dwell-based methods. This lower performance can be partially attributed to the use of a non-QWERTY keyboard layout and the lack of word prediction, requiring users to spend additional time adapting to the keyboards design and type the entire word. Overall, the results suggest that while adaptive control can enhance stability and precision, it often does so at the expense of typing speed. More recently, Multi-Threshold Dwell [29] achieved a much higher average of 18.3 WPM, with a moderate error rate of 2.0% due to shorter dwell durations and increased typing speed. It also scored well in comfort, with an average rating of 4.5, showing that well-tuned dwell thresholds can significantly improve efficiency while maintaining usability. Finally, GazeTheKey [39] performs slightly lower with an average speed of 9.34 WPM. The main drawback of most dwell-based methods is user fatigue and boredom caused by waiting for selections to trigger, reflected in comfort levels averaging 3.5.

In contrast, Dwell-Free methods generally exhibit higher error rates, primarily due to the Midas touch issue, where unintended selections occur. For example, Filteryedping [34] and GazeTry [23] show average error rates of 4.5% and 4%, respectively. However, HGaze [14] effectively mitigates this issue by allowing users to confirm selections with a head nod rather than relying on gaze fixation alone. This method achieved an error rate as low as 0.37% by the final session, making it one of the most accurate among Dwell-Free approaches. In terms of typing speed, Filteryedping leads with an average of 15.95 WPM, reaching 16 WPM in the last session, outperforming HGaze's average of 11.22 WPM. Nonetheless, HGaze has good comfort ratings, with an average score of 4.8, suggesting it as a viable option for longer use due to its user-friendly design. pEYEwrite [21], despite lacking predictive text or feedback mechanisms, still achieved 12.33 WPM and an exceptionally low error rate of 0.01%, demonstrating that some designs can perform well even without assistive aids.

Meanwhile, Pursuit-based techniques, such as Mobile Gaze Interaction [30], Leyenes [11], SPEye [36], and SMOOVS [33], offer a calibration-free alternative that enhances robustness at the cost of speed. Mobile Gaze Interaction reported slower typing rates than dwell-based systems but benefits from easy setup and calibration-free operation. Leyenes achieved an average of 1.32 WPM with very low error rates, while SPEye reached 1.02 WPM and an error rate of 0.01%, confirming pursuit-based systems as highly reliable but slow options and fatiguing over time. SMOOVS, with its distinctive hexagonal layout divided into six triangular regions, reached 3.34 WPM, outperforming most other pursuit-based methods and showing that spatially optimized layouts can partially compensate for pursuit tracking limitations. These systems are ideal for users who cannot maintain stable gaze calibration over time, favoring reliability over typing speed.

Unconventional keyboards also benefit greatly from predictive and feedback features, as users are often unfamiliar with their layouts and functions. This initial unfamiliarity may lead to higher error rates, which generally decrease as users adapt. For example, Dasher [22] initially showed an error rate exceeding 10% in the first session, which dropped below 5% by the end as users became more accustomed to it, resulting in an overall rate of 4.4%. Its typing speed also increased from 9.3 WPM to 19.2 WPM however the tests were conducted using a mouse instead of eye tracker so real speeds are expected to be lower. One-Dimensional Eye-Gaze [9] and WordPop [10] avoided this learning curve due to their simplicity, achieving average error rates of 1.26% and 0.19% (for Reverse Crossing), and typing speeds of 11.36 WPM and 5.41 WPM, respectively. Comfort tends to be higher for unconventional layouts that allow steady or minimal gaze movement, such as One-Dimensional Eye-Gaze, which reached a comfort score of 6, and WordPop, with 5.46 using Dwell Time and 4.93 with Reverse Crossing.

Overall, Dwell-based methods remain the most accurate and well-balanced in terms of speed and comfort, especially when enhanced with adaptive or multi-threshold mechanisms. Pursuit-based methods excel in robustness and accessibility for users unable to perform precise fixations, at the cost of much slower speeds. Unconventional keyboards strike a middle ground, often trading initial learning effort for long-term comfort and fluid gaze interaction. Ultimately, the ideal

technique depends on the users needs, whether prioritizing speed, accuracy, or robustness over calibration and fatigue.

Technique	Eye Tracker	Selection Method	WPM	Error Rate (MSD)	Comfort (1-7)	Participants and Sessions	Word Prediction	Visual or Auditory Feedback
Adjustable Dwell Time [26]	Tobii 1750 (0.5°)	Dwell Time	Start:6.9 End:19.9 Avg:15.7	Start:1.28 End:0.36 Avg:0.82	3.5	10/10	No	Yes
Dynamic Dwell Time [35]	Tobii X60 (0.5°)	Dwell Time	Start:21.3 End:16.6 Avg:18.4	Avg:<1	N/A	10/8	Yes	Yes
GazeTheKey [39]	SMI REDn (0.4°)	Dwell Time	Avg:9.34	N/A	N/A	10/5	Yes	Yes
One-Dimensional Eye-Gaze * [9]	Tobii Eye Tracker 5 (1°)	Dwell Time	Avg: 11.36	Avg: 1.26	6	8/(N/A)	Yes	Yes
GazeTry [23]	TheEyeTribe (0.5-1°)	Dwell Free	N/A	Start:5 End:2.5 Avg:4	N/A	2/5	Yes	No
Filteryedping [34]	Tobii REX (<0.6°)	Dwell Free	Start:11.7 End:16 Avg:15.95	Start:6 End:4 Avg:4.5	N/A	12/3	Yes	Yes
ContextSwitching [15]	Low Cost eye tracker (no name disclosed)	Saccade	Start:6.5 End:13.3 Avg:12	Start:15.6 End:5.8 Avg:<8	3.97	6/8	No	Yes
Dasher * [22]	Mouse	Gestures	Start:9.3 End:19.2 Avg:17.3	Start:>10 End:<5 Avg:4.04	N/A	10/6	Yes	Yes
HGaze [14]	Tobii Eye Tracker 4C (2-3°)	Dwell Free Noding	Start:7.3 End:12 Avg:11.22	Start:<3.5 End: 0.37 Avg:N/A	4.8	10/9	Yes	Yes
pEYEwrite * [21]	Eyelink2 (0.25- 0.5°)	Saccade	Avg:12.33	Avg: 0.01	N/A	4/5	No	No
WordPop * [10]	Tobii Eye Tracker 4C (2-3°)	Dwell Time/Saccade	Avg: 5.5/4.86	Avg: 0.57/0.19	5.46/4.93	15/2	Yes	Yes
Mobile Gaze Interaction [30]	IphoneX camera	Pursuit	Avg: N/A	Avg: N/A	N/A	24/1	Yes	Yes
Leyenes * [11]	Gazepoint GP3 HD eye tracker (0.5°-1°)	Pursuit	Avg: 1.32	Avg: N/A	N/A	24/1	Yes	Yes
SPEye * [36]	Tobii Eye Tracker 4C (2-3°)	Pursuit	Avg: 1.02	Avg: 0.01	N/A	3/22	Yes	Yes
Adaptive Dwell Time * [19]	The Eye Tribe (0.5-1°)	Dwell Time	Avg: 1.93	Avg: N/A	N/A	7/3	No	Yes
SMOOVS * [33]	SMI RED-oem eye tracker (0.5°)	Pursuit	Avg: 3.34	Avg: N/A	N/A	24/1	Yes	Yes
Multi-Threshold Dwell [29]	Meta Quest Pro VR headset (1.652° [44])	Dwell Time	Avg: 18.3	Avg: 2.00	4.5	15/5	Yes	Yes

Table 2.1: Technique Statistics: 1) Techniques marked with * are unconventional, meaning they do not use a standard QWERTY keyboard layout. 2) Comfort Levels are rated on a scale from 1 (very uncomfortable) to 7 (very comfortable). 3) Participants and Sessions dictate the count of participants, followed by the number of sessions, represented as (participants/sessions). 4) N/A indicates insufficient data available for a particular measure.

Chapter 3

HexGaze Text Entry Technique

In this chapter we describe our HexGaze technique covering its layout and selection methods as well as how the feedback is provided to the user after selection. Then, we explain how the word prediction works and how to use it correctly. Finally, we describe some software specifics of the virtual keyboard and finish it up with a summary.

3.1 HexGaze Technique

As we saw in the previous chapter, some existing techniques can be uncomfortable or exhausting, often leading users to feel fatigued or bored after prolonged use. To overcome this we need a method that minimizes extensive eye movement and user fatigue, while keeping typing speeds high. To that end, we created Hexgaze, a virtual keyboard composed of self centering hexagons divided into segments that contain words and letters. To select them the user has two diferent selections methods: Gesture selection and Dwell Time selection. When selecting the user will receive auditory and visual feedback in the form of progress bars and beeping sounds. To speed up the process the virtual keyboard contains letter and word recommendations according to what the user has typed and is typing. It allows users to select an entire word or type it letter by letter while keeping the gaze mostly in the center of the screen [5]. This avoids rapid and wide movements, while keeping the selection zone where the eye tracker has the highest precision.

3.2 Virtual Keyboard Layout

The HexGaze user interface is designed for clarity and efficiency, presenting a clean, symmetric and intuitive environment for text entry. The layout is divided into two main sections: a status display at the top and an interactive virtual keyboard canvas at the center, where the gaze is normally directed at [49].

As shown in Figure 3.1 at the very top of the screen (1), two text fields provide essential real-time feedback. The first field displays the sentence that users should type, and the second field shows the text typed so far. This side-by-side comparison allows for immediate self-correction and progress tracking without needing to look away from the center of the screen.

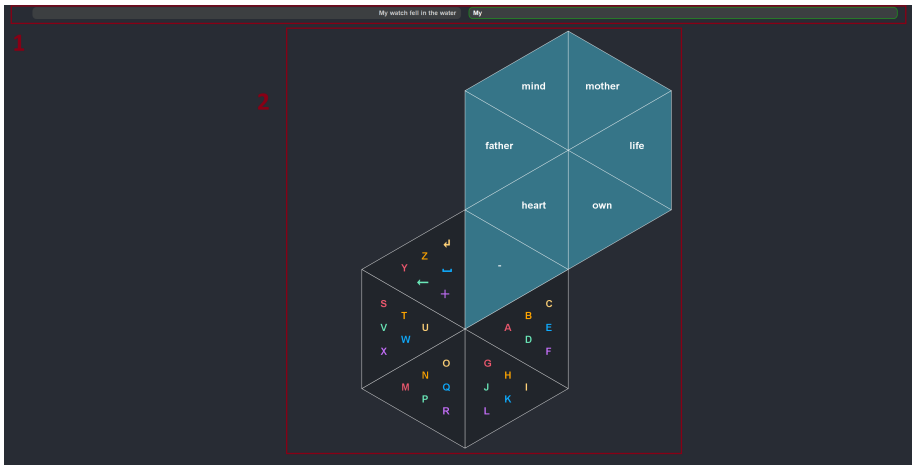


Figure 3.1: HexGaze Layout, showing the two main sections: 1) status; 2) virtual keyboard (Initial Group Hexagon).

The core of the application is its dynamic keyboard, which occupies the central canvas (2). Since eye trackers have less precision on the edges of the screen we tried to keep everything in its center, making it easier on the eyes and avoiding areas where the eye trackers have less precision. Instead of a traditional grid of squares, the keyboard is constructed from an interconnected web of hexagons.

The keyboard is composed of interconnected hexagons, allowing smoother navigation between elements, facilitating gesture-based interaction, and saving screen space. We chose a hexagonal layout for the virtual keyboard because hexagons connect more naturally with one another and reduce travel distance compared to square or circular layouts [7]. Unlike other grid shapes, hexagons preserve both linearity and directionality, a property that is essential for accurately representing spatial relationships and movement patterns. Prior studies have shown that hexagonal grids provide greater efficiency and accuracy than alternative grid types [1]. Furthermore, hexagons are the most space-efficient tiling shape, enabling a compact and ergonomic layout without gaps. This geometry also supports smooth swiping motions, as each hexagon is equidistant from its neighbors, which facilitates easier transitions between letters [4]. Finally, the hexagons natural division into six segments offered an effective way to group letters, providing sufficient space to accommodate the entire alphabet and additional symbols.

Each hexagon is divided into 6 segments, with each segment containing a set of 6 coloured letters, logically grouped in alphabetical order for ease of search, guiding the user to the next likely character.

Among the letters are also dedicated action keys, to which we have given special emphasis in the design:

- **Delete Letter (←):** This key, marked with a backspace arrow, performs the standard function of deleting the last character typed. It is the primary tool for correcting minor errors one character at a time.

- **Delete Word (↲):** A more powerful correction tool, this key is designed for greater efficiency. When selected, it deletes the entire last word typed. This allows for much faster editing, as a single selection can remove a complete mistake, saving the effort of repeatedly deleting individual letters.

- **Space (␣):** The space key functions as expected, inserting a space to separate words. Its selection signifies the completion of a word, which in turn allows the system to offer predictions for the next word.

- **Plus (+):** This key switches the letter keyboard to a number and punctuation keyboard, allowing the user to type more well-formed phrases with proper punctuation.

Adjacent to the group hexagons blue segment is a word list hexagon of the same color. This hexagon displays the top six most likely words based on what the user has typed and is currently typing. It uses a bigram language dataset to predict the next possible words based on the previous word, and a word frequency dataset to predict the word the user is typing based on currently typed letters.

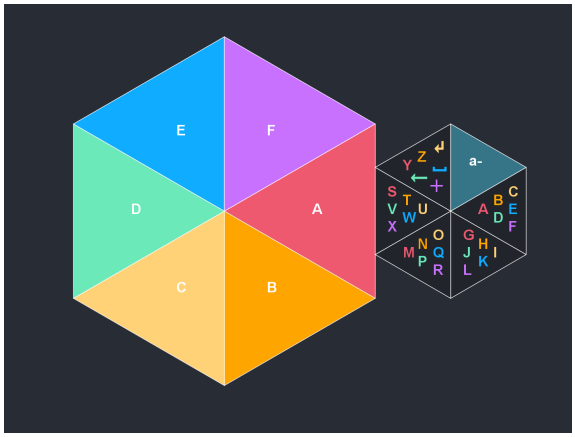


Figure 3.2: HexGaze Layout (Letter Hexagon).

After the users go through all the selection processes to select a segment with the intended letter in the group selection hexagon they are presented with a new letter hexagon layout (Figure 3.2). This layout consists of a hexagon divided into six segments, each containing one of the letters from the previously selected segment. The background colour of each segment matches the colour of the corresponding letter in the previous hexagon, allowing the user to quickly and easily locate the desired letter by color using peripheral vision [31]. So in summary we contain three different hexagons: A group hexagon, which contains groups of letters in each segment, a letter hexagon which contains individual letters in each segment from the group hexagons corresponding adjacent segment, and a word list hexagon, which contains the predicted words.

The segments in all hexagons are numbered clockwise from 1 to 6. The first segment is the one at the right-center position (which contains the first six letters of the alphabet in the group

hexagon in Figure 3.1), and the last is the top-right segment (which contains the blue triangle in the same figure).

Although a clockwise numbering would normally start from the top-right, we chose to begin at the right-center instead. This choice centralizes the placement of letters and positions words in a less precise area. Since selection in that spot is dwell-based only, the risk of accidental selections is reduced.

Another advantage of starting at the right-center is that predicted words also begin from this position. This places the four most common words out of the six predicted words in the lower half of the hexagon, closer to the center, which facilitates search and selection. Only in rare cases will a user select a predicted word at the very top (the last of the six).

The hexagons radius is set to one-third of the screen height, making it large enough to compensate for eye tracker imprecisions, while still small enough to allow multiple hexagons to fit on the screen without going out of bounds. Each segment within the hexagon is of equal size and angles, ensuring visual consistency and aesthetic appeal.

These keys are also color-coded with bright and different colours, allowing users to locate them more easily and distinguish them from far away without losing sensitivity to them given that green-red colours become blind to the users above 30 degrees, while blues and yellows only above 60 degrees[47][17]. Additionally, the same colour is used as the background for the corresponding segment in the next hexagon, helping users quickly identify the target letter through colour association.

The letters are positioned in alignment with the triangular segments vertices, maintaining perfect symmetry and making the layout visually pleasing and easy to navigate.

To save screen space, the hexagon was rotated to a pointy-top orientation instead of the traditional flat-top orientation [1]. This allows more hexagons to fit horizontally, where the screen is widest, than vertically.

The background of the app group hexagons segments is a dark color and the font size is bigger to reduce the digital eye strain (DES) that has been increasing with the use of computers making the users eyes less tired and able to use the app for prolonged times [8].

3.3 Interaction and Selection Mechanism

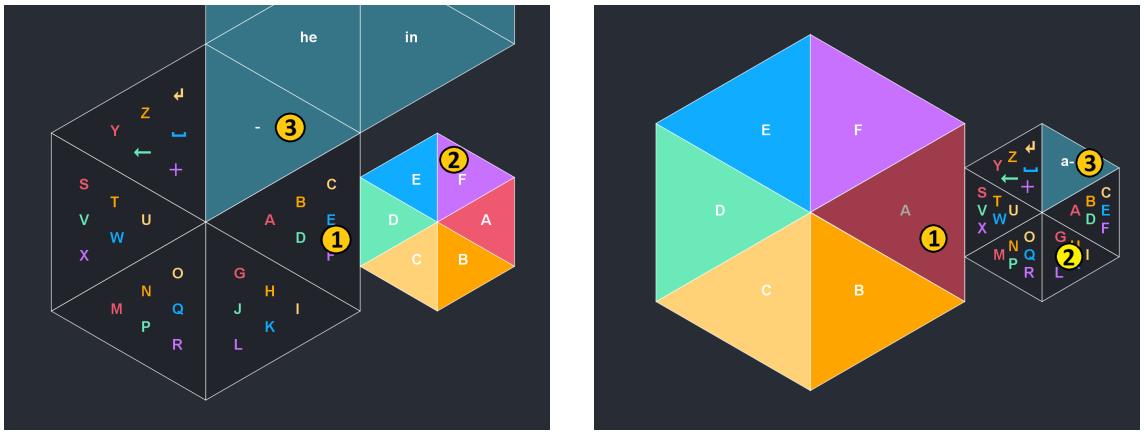
The selection mechanism is a valuable part of any eye-typing technique, directly influencing its speed, accuracy, and overall user comfort. A robust system must therefore provide versatile selection methods to different user needs and contexts. For rapid, continuous typing, a fast and fluid gesture-based approach is ideal, allowing users to string characters together with minimal interruption. Conversely, for moments requiring deliberate precision or for users who may find gestural control challenging, a more traditional dwell-time method offers an essential, reliable alternative. Our HexGaze virtual keyboard combines these two complementary approaches to create a hybrid and versatile text entry solution.

[1] https://www.redblobgames.com/grids/hexagons/

3.3.1 Gesture Selection

Gesture-based selection is the main selection method, optimized for speed and fluidity.

When writing a word, the users focus their gaze on the triangle of the group hexagon that contains the desired character (Figure 3.3a-1). The system registers this area of interest. As we can see in Figure 3.3a after gazing at the segment containing the first six letters of the alphabet a new smaller hexagon appears adjacent to it (Figure 3.3a-2). This is how we will change hexagons. In order to shift, the users need to gaze at this new hexagon and by doing so a smooth animation will be played converting the smaller hexagon (Figure 3.3a-2) into a bigger one while centralizing it on the screen (Figure 3.3b-1).



(a) Initial State Selection.

(b) Letter Selection.

Figure 3.3: Gesture Selection: (a) Initial State and (b) Selecting Letter.

When the hexagon reaches it maximum size and the center of the screen, the users are able to type. To select a specific letter, the users repeat the initial gesture, gazing at the segment containing the desired letter (eg:'A')(Figure 3.3b-1) and a new group hexagon appears adjacent to it (Figure 3.3b-2). This new hexagon is similar to the first one except that it takes into account the letter the users are gazing at, thus displaying it in the blue triangle (Figure 3.3b-3) and only showing letters that can occur after the already written. The system detects this gaze transition from the letter triangle to the neighbouring hexagon as a single, cohesive gesture. This gesture confirms the selection of the letter that was in focus, playing a confirmation sound and simultaneously moving the center of interaction to the newly selected hexagon, preparing the system for the next text entry.

This method transforms writing into a sequence of continuous movements, eliminating pauses and significantly reducing the time required to select each letter.

Rapid and constant on-screen movements can cause significant eye strain over extended periods, as they force the user to constantly readjust their focus to new positions [41]. To mitigate this, we designed an animation strategy that minimizes abrupt, long-distance transitions. Our solution involves two key adjustments: first, we reduced the radius of adjacent hexagons to half of the central one. This change brings the next target hexagon closer to the screens center, significantly shortening the eyes travel distance. Second, as the new hexagon moves into focus, it simulta-

neously grows in size. This scaling effect further minimizes the perceived camera movement, resulting in a smooth and centralized transition that is less demanding on the eyes.

To further mitigate eye strain, we ensured that all animations render at a high frame rate, targeting above 60 frames per second (FPS). Lower frame rates can introduce perceptual artifacts such as stuttering and motion blur, which force the eyes to work harder to track moving objects and can lead to increased fatigue. By maintaining a high and stable frame rate, we provide a smooth and clear visual experience, making the interface more comfortable for prolonged use [43].

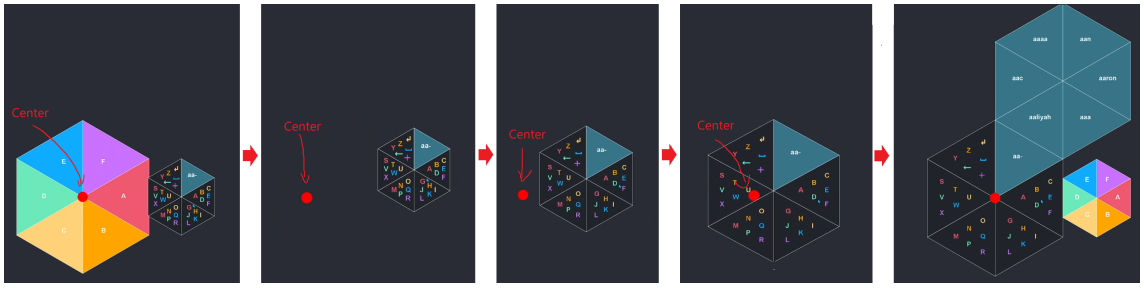


Figure 3.4: Animation to center the focused Hexagon.

In Figure 3.4 we can see a representation of the hexagon recenter animation, where the red dot represents the center of the screen with the hexagon moving towards it while increasing in size. After some preliminary tests we found the best time for the animation to be 800 ms. This was the most comfortable balance between speed and usability. Longer durations would slow down typing, while shorter ones would make the animation too rapid, causing disorientation on the users and increasing eye fatigue. At around 70% of the animation the users are allowed to type again to speed up typing.

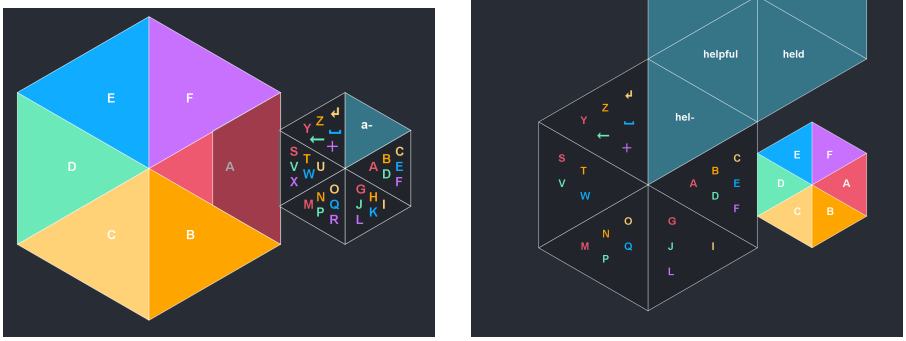
3.3.2 Dwell Time Selection

As an alternative, the system offers a selection mode by dwell time, which functions as a support mechanism and ensures accessibility. This method is specifically designed for selecting from the letter and word prediction hexagons. To activate these options, the users must first perform a directional gesture to bring the letter hexagon into focus (same as before). Once the letter hexagon is in the center, the users must hold their gaze on the specific letter or word they wish to select. A discreet timer starts, and if the gaze remains fixed for a designated duration the letter or word is selected. For letters the dwell time is 500 milliseconds and for words 700 milliseconds. The latter value is slightly longer to give users time to read the words. These values were selected after some preliminary tests.

This selection mode, although inherently slower, is fundamental for providing a reliable alternative for interaction. The coexistence of the fast gestural method and the deliberate dwell-time method creates a hybrid and versatile text entry system, capable of adapting to different user needs and preferences by prioritizing comfort and speed without sacrificing accessibility.

3.4 Feedback

For an eye-typing system to be effective, it must provide clear feedback to build user confidence. Our interface uses both visual and auditory cues to confirm the users gaze location and signal every successful input.



(a) Dwell Selection Feedback, showing 70% progress

(b) Group Hexagon Text Feedback.

Figure 3.5: Hexagons Feedback Mechanisms: (a) Dwell Time Feedback and (b) Currently typed word Feedback (word “hello”).

For the dwell-time selection method, since it is based on a fixation interval, visual feedback is provided via a transparent progress bar that appears over the target triangle and grows from the outside to the inside of the hexagon (Figure 3.5a). This clearly indicates how much time remains until the selection is confirmed allowing users to either stay and confirm or look away without selecting. To confirm that an action was successfully completed such as inputting or deleting a letter or word, the system provides auditory feedback via a “beep” sound that is played, allowing users to understand their actions.

In the group hexagon, in addition to the segments containing letters, there is one dedicated segment reserved for displaying the letters already typed of the current word. As we can see in Figure 3.5b below the word list hexagon there is a blue triangle with a “hel-”. This shows users what they have currently typed of the present word, avoiding the need to look up and down to check for mistakes. When there is no letter written for a word, a “-” is displayed (Figure 3.3a-3). When the typing begins that character is replaced by the currently typed letters, maximizing efficiency. As seen in Figure 3.5b the user is currently typing the word “Hello” but only “Hel” has been typed so far, being shown in the triangle. This way the user can receive real time feedback and keep up with what is being typed while avoiding fast movements to the top, improving speed and reducing accidental selections caused by those movements.

For our timed user tests, we needed a clear method to show participants when the timer started and stopped and when they could type. We achieved this by adding a border around the text field,

providing an intuitive visual signal for the timers active state [10]. The system communicates the input state through the main text field: a green border indicates that the user is able to type and the time is running (Figure 3.6a), while a red border signals that input and timer are temporarily disabled, giving users the time needed to read the phrase they are going to type without any accidental selections (Figure 3.6b).

(a) Text Field Feedback Permission Granted

(b) Text Field Feedback Unable to Write

Figure 3.6: Text Feedback mechanisms: (a) Permission Granted and (b) Permission Deny.

3.5 Suggestion Mechanism

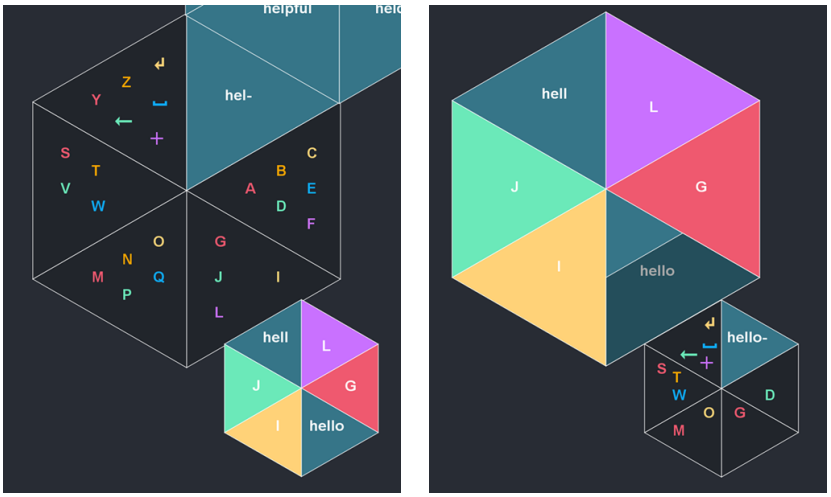
Another key implementation to improve performance is autocompletion, where the system predicts what the user intends to type and offers a selection of words. To accelerate text entry, the keyboard integrates a dual-engine suggestion mechanism based on the word prediction and the next most probable word after the last written one.

3.5.1 Current Word Prediction

The first engine provides in-word predictions, offering to complete the current word as the user types. This functionality is driven by a frequency-based model, which analyses a unigram dataset based on the top 333 333 most frequent words in the English Language [2] to suggest the most common words that match the current prefix.

The six most likely word suggestions are presented in the word list hexagon, allowing the user to select one quickly by dwelling on it (Figure 3.5b). Selecting words from this hexagon will automatically add a space after to allow users to begin typing the next word right away and speed up the process. We also included a dynamic filtering mechanism for presenting words beside those on the list. As the user types a prefix, the keyboard removes any characters that cannot be appended to that prefix to form a valid English word. This is demonstrated in Figure 3.7a, where after the user typed “Hel”, the system removes invalid next letters. In this case the letters “H” and “K” were removed from the keyboard in both the group hexagon segment and the following letter hexagon, guiding them more efficiently toward their intended word, “Hello.”

[2] https://www.kaggle.com/datasets/rtatman/english-word-frequency



(a) Keyboard State and Removed Letters With "Hel" Typed.

(b) Letter Hexagons Filled Spaces with Autofilled Words.

Figure 3.7: Current word prediction mechanisms: (a) Removing Letters and (b) Adding words to removed letters space.

After removing the letters we used the blank spaces to present autocompleted words. Depicted in Figure 3.7b is a Letter Hexagon missing the letters "H" and "K", and in their place are the two most common words based on the prefix "Hel" that contain any of the letters presented in this hexagon, which are "L", "G", "J", and "I". Thus when the user gestures to this hexagon the system takes into account all the possible letters inside it and finds the most common words containing the prefix plus the letters inside. In this case there are four letters and two spaces so it shows the two words with most frequency of all the available letters.

Unlike the main word list, these suggestions can be selected by either gesturing or dwelling. A space is not automatically added, which is useful when a suggestion is a prefix of the target word. For example, a user can select the word "Close" and then immediately be presented with the letters needed to finish the word "Closed", giving more efficiency.

3.5.2 Next Word Prediction

The second engine focuses on next-word prediction. After a word is completed, this system suggests a list of likely subsequent words. This predictive capability is powered by a more advanced bigram[3] language model, which calculates the probability of the next word based on the word that was just entered.

To help the user write more quickly, the system predicts the next word they are likely to need. This is handled by a bigram language model that knows which words tend to follow others. At the beginning of a sentence, the keyboard suggests the six most common starting words (Figure 3.3a).

[3]https://www.kaggle.com/datasets/dm4006/google-n-gram/data?select=wp_2gram.txt

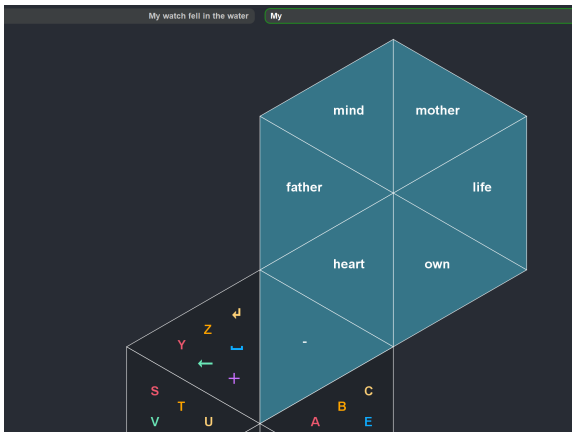


Figure 3.8: List of Predicted Words After “My”.

As soon as the user types a word, such as “My”, the suggestions instantly adapt, showing the six words most likely to come next in the word list hexagon (Figure 3.8). This allows the user to simply select a word to continue [4]. The system counts a word as finished when there is a space after the word. If the space is then deleted the system returns to the current word prediction. Selecting from these words also add a space after selection to allow to also predict next word after the selected one, making a chain of next word predictions.

3.5.3 Next and Current Word Prediction

In cases where the user finishes typing a word and overlooks the next word prediction list, we integrate both engines. Specifically, if the users begin typing the next word instead of selecting a predicted one, and the word they are typing was already present in the previous prediction list, it is immediately displayed without requiring the user to wait for it to reappear.

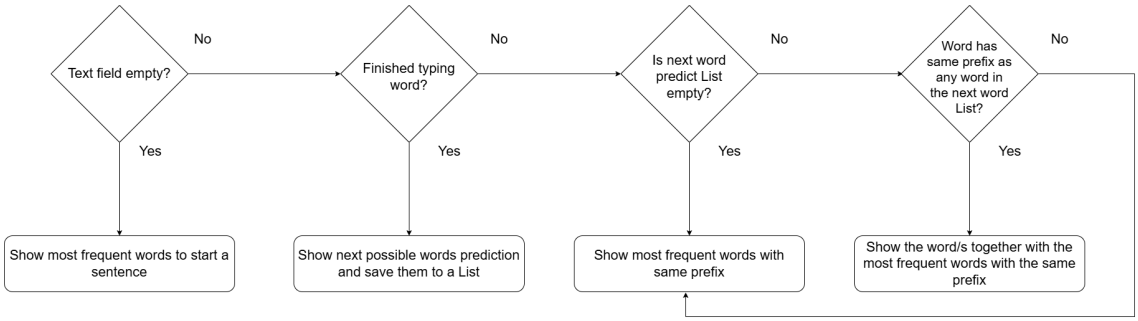


Figure 3.9: Both Prediction Methods Diagram.

In Figure 3.9, we illustrate how the prediction system works. At the beginning, the text field is empty, so the next-word prediction shows the most common words that typically start a sentence. Once the user begins typing, the system switches to current-word prediction. Since this is the first word, the next-word prediction list remains empty. After the first word is completed, the system generates the next possible words and stores them in a list.

[4]The underlying language model data is a subset of a very large file to ensure the app runs efficiently.

If the user ignores these suggestions and starts typing a new word, the system compares the typed prefix with the saved next-word predictions. If the typed prefix matches one of these predictions, the corresponding words are displayed. The remaining slots are then filled with current-word predictions. This ensures that even if the user skips the next-word suggestion initially, the predicted word can still be quickly selected without additional typing.

Because of the large size of the original bigram model, we created a smaller version by limiting storage to six next-word predictions per word to reduce resource usage. As a result, only those six words are able to be displayed, and no additional candidates with the same prefix are shown.

To use these features effectively and maximize typing speed, the user should typically type two to three letters of a word before selecting it from the word list. For common short words and pronouns, typing a single letter is often sufficient, whereas longer or less common words may require more letters, particularly when many words share the same prefix. After finishing typing a word it is recommended to always check for next predicted words from the word list as selecting them can speed up the process and maximize comfort. Another recommended technique is to take advantage of cases where after finishing a word, the predicted next word is a prefix of the intended word. For example, if the user wants to type “Universities” but the system predicts “University”, the user can select the predicted word and then delete the trailing space and the letter “y”. This action reactivates the current-word prediction, which will display all candidate words related to “Universit”, including the intended word. In case the user intends to type a word that does not exist in the English dictionary the system will not find any possible matches and display the normal keyboard without any autofill mechanics.

3.6 Implementation Details

The application was developed in Java using the Swing framework, a choice driven by our familiarity with the language and past experience with drawing in JFrame. To ensure a responsive interface, the UI is built upon a JLayeredPane, which divides the screen into three independent panels. The first is the Top Panel, which contains both text fields at the top. The second is the main Drawing Panel, where the hexagonal keyboard is rendered. Finally the Progress Panel, that handles the dwell time progress bar.

This separation is key to optimizing the applications speed. Instead of repainting the entire screen for every small change, only the relevant panel is redrawn. When a letter is added, only the Top Panel updates and is repainted. When a dwell selection is in progress, only the small progress bar is animated, without forcing the complex hexagons to be redrawn repeatedly. Because the drawing panel is only drawing the hexagons it is able to draw the centering animation, without stuttering. This approach prevents rendering bottlenecks and keeps the application fluid.

Another optimization was to pre-load all Color and Font objects at startup. Repeatedly creating new colors and fonts inside the drawing code can cause the application to stutter, because Java’s garbage collector must pause the program to clean up the unused objects. By defining all colors and fonts as constants and reusing them, we prevent these pauses and ensure a smooth animation.

Word Datasets can be very expensive on resources, specially for loading times and memory consumed. So we did some optimizations in order to implement the word predictions using two distinct datasets. While the dataset for in-word prediction was smaller than our next word bigram model, its 333,000 words still posed a performance challenge. Our initial implementation relied on a linear search to find matching prefixes, an approach with a time complexity of $O(N * M)$ (where N is the number of words and M is their average length), which was too slow for real-time feedback. To solve this, we changed the system to use a Trie data structure. This allows for prefix lookups in $O(L)$ time, where L is simply the length of the prefix itself, enabling instantaneous word suggestions and significantly improving application performance.

The bigram dataset required a different optimization strategy due to its immense size. Our initial attempt to load the entire dataset into memory was unsuccessful, consuming roughly 16 GB of RAM and requiring 40 seconds to initialize the app. To overcome this, we developed a filter to parse the original 99-million-line file with specific criteria: for each word, we kept only its top six subsequent words, and we ensured these words were purely alphabetic English words, no commas or abbreviation. This filtering reduced the dataset from 99 million lines to just 1.6 million. As a result, memory consumption dropped from 16 GB to 1.5 GB, and load times plummeted from 40 seconds to under half a second, keeping the app viable and fast for use.

3.7 Summary

We described our hexagonal virtual keyboard designed for comfort, speed and to support low-cost eye trackers by keeping the keyboard positioned in the center of the screen. This is where tracking accuracy is higher as well as where users tend to gaze more frequently, making interaction more comfortable. The keyboard is composed of interconnected hexagons, allowing smoother navigation between elements, facilitating gesture-based interaction, and saving screen space. It begins with a group hexagon divided into six segments, each containing a set of six alphabetically ordered letters except for one segment reserved for functions. Gazing at a segment reveals an adjacent letter hexagon displaying the letters from that group, distributed across its six segments. Two selection methods are available: gesture-based selection where user gazes towards the next hexagon or dwell-based selection by fixating on a letter for a set period of time. To select a letter, the user first gestures towards the corresponding letter hexagon and then either dwells on the desired letter to return to the initial state or gestures towards another group hexagon adjacent to the letter being viewed. A predicted word list is always displayed next to the group hexagon, updating dynamically based on the text typed so far, with predictions selectable by dwelling. As typing progresses, “impossible” letters disappear and are replaced by additional word predictions. Selecting a predicted word automatically inserts it followed by a space. The keyboard also includes a progress bar for dwell selections, an audible confirmation beep for all selections, a delete word button, a delete letter button, and a space button integrated into one of the letter segments.

Chapter 4

Experimental Evaluation

In this chapter, we present the two studies conducted to evaluate our application: one comparing our solution to three other virtual keyboards, and another examining how users' performance evolves over time while using our technique. We then analyse the results collected in these studies, focusing on text entry rate (WPM), error rate (MSD), keystrokes per character (KSPC) and participants' feedback. Both studies received approval from the faculty Institutional Review Board.

4.1 Study I - Comparison with Other Techniques

In this section, we analyse how our technique compares with two others, WordPop [10] and Dasher [22]. We compare them in terms of speed, comfort, error rate, and user enjoyment. This experiment was conducted to determine which technique participants preferred, felt more comfortable and performed best with.

4.1.1 Participants

We recruited 21 participants by word of mouth for this experiment, comprising 11 males and 10 females, aged between 15 and 56 with an average of 25.7 (SD = 11.15), all without disabilities and with no prior experience using eye tracking. Among them, eight wore glasses or contact lenses, none were native English speakers, and two reported limited proficiency in English.

4.1.2 Apparatus

The tests were conducted on a desktop computer equipped with a i9-14900K CPU at 3.2 GHz base speed and 32 GB of RAM, running Windows 11, and connected to a 28-inch monitor with a resolution of 3840 × 2160. Eye tracking was performed using a Tobii Eye Tracker 4C, mounted underneath the monitor with participants standing 60 cm in front of it. All tests where conducted in a quiet, well lit and calm place with only the presence of the investigator and the participant. All tests were in the same place.

4.1.3 Experimental Procedure

The study session for each participant consisted of three parts, each evaluating a different technique. A total of 32 phrases were selected from the MacKenzie and Soukoreff dataset [24], ensuring a balanced mix of simple phrases, complex phrases, and phrases containing many pronouns and short words. These were divided into four groups of eight phrases each: one group for training and one group for each part (technique). The words groups were numbered from 1 to 3 and remained fixed to their respective part, meaning that in the first part, all participants always typed the phrases from group 1 and so on. We used a within-subject design.

We began the experiment by explaining to each participant the purpose and objectives of the study, what they were doing, and clarifying that it was the techniques being evaluated rather than the participants themselves. We then asked them to sign a consent form and to fill a personal form about their age and eye sight issues as well as if they use glasses or contact lenses.

Before starting, we calibrated the eye tracker for each participant and only began the typing sessions once we confirmed that the tracked gaze closely matched the participants actual gaze. While a small margin of error was always present, it was generally close to the intended point of focus. If participants reported that the gaze estimation was inaccurate, we recalibrated the eye tracker until they were satisfied and able to perform well.

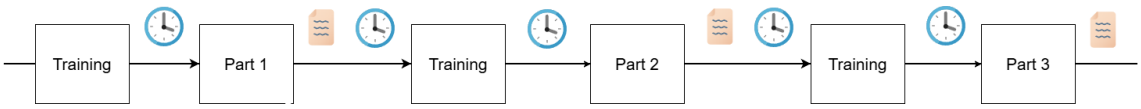


Figure 4.1: Steps of the experimental procedure of study I (The clock icon represents a 10-minute break and the paper icon indicates the questionnaire phase).

When calibration was successful we began the typing sessions, in Figure 4.1 we can see the timeline of events of each participants session. Before starting each part, users had a period of around 10 minutes to practice typing using the technique correspondent to that part. After practice, participants had a 10-minute break and then were shown the phrase to type, for them to memorize. The phrase was kept visible on the screen, but the procedure encouraged participants to type in a more natural manner, similar to when one writes from memory rather than constantly rereading. Nonetheless, the text was available at all times for any quick reference. Once they were ready we pressed a key on the keyboard to allow them to begin typing. When done typing that phrase we then pressed the same button to finish the typing of the sentence. This pause allows users to rest and look at the new phrase, which is shown when the pause starts. The system recorded timestamps for each typed character and calculated typing speed using the moment the last character was entered, ensuring that pauses after finishing a sentence did not distort the measurements. This process repeated until the last phrase was completed.

After typing all eight phrases in the technique, participants were asked to complete a questionnaire about that technique. The form included questions regarding comfort, eye fatigue, the perceived difficulty of learning the technique, ease of typing with it, and perceived typing speed.

All questions were answered using a 7-point Likert scale, ranging from 1 (very poor) to 7 (excellent). Then, we gave participants a 10 minute break before moving onto the next part with a different technique.

This process was repeated three times until each participant had typed eight phrases with all techniques. At the end, they completed a final questionnaire asking which of the three techniques they preferred and providing any suggestions for improving our technique.

To eliminate any potential biases, we applied a Latin Square design to rotate the sequence of techniques for each participant. For example, since the word groups are fixed to the parts if one participant used Dasher first, followed by WordPop and then HexGaze, the next participant would begin with WordPop, proceed to HexGaze, and finish with Dasher. This ensured that each technique was tested with all word groups and prevented any technique from being associated with an easier set of phrases.

4.1.4 Experimental Results

After completing the tests, we gathered the data and performed statistical analyses to analyse the results of the experiment, including typing performance (WPM), error rate (MSD), keystrokes per character (KSPC), comfort, and user feedback. For the performance data, since it is of type ratio, we used the Shapiro–Wilk test to assess normality. Based on the normality test, we then selected the appropriate statistical test to determine whether there were significant differences between the three techniques. For the feedback metrics, since they are ordinal and involve three conditions, we used the non-parametric Friedman test to identify potential differences. Due to poor eye-tracking accuracy when using glasses, one of the participants data was removed, as the test could not be completed properly and the typing performance deviated significantly from the rest.

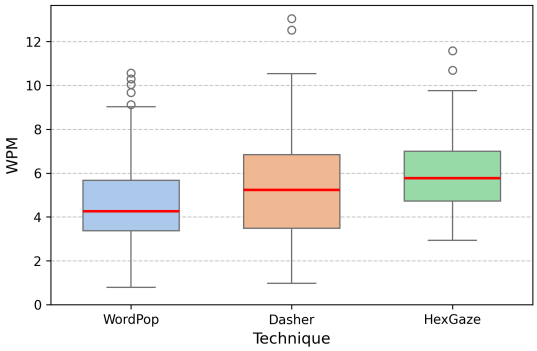
Table 4.1 shows the results of the Shapiro-Wilk test for each metric and technique, which will be discussed in more detail below. For the KSPC metric, WordPop's data was excluded because, upon analysis, we noticed irregular results. Further investigation revealed that the WordPop application was counting incorrectly the keystrokes, which compromised the validity of the results.

Table 4.1: Shapiro–Wilk test results and statistical comparison of the three techniques for performance metrics.

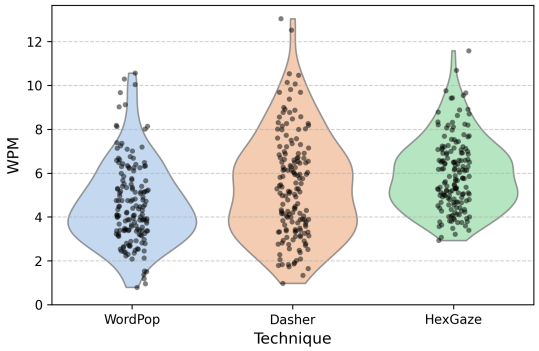
	WPM			**MSD (%)**			**KSPC**		
Technique	*W*	*p*	*Norm*	*W*	*p*	*Norm*	*W*	*p*	*Norm*
WordPop	0.938	0.216	True	0.943	0.269	True	—	—	—
Dasher	0.973	0.820	True	0.835	**0.003**	False	0.948	0.333	True
HexGaze	0.923	0.112	True	0.813	**0.001**	False	0.961	0.566	True
	RM-ANOVA			Friedman Test			Paired T-test		
	$F(2, 38) = 6.79, p =$ **0.003**			$\chi^2(2) = 1.6, p = 0.449$			$T(19) = 13.87, p <$ **0.001**		
	$\eta^2_G = 0.164$			$W = 0.04$			$d = 4.32$		

Text Entry Rate

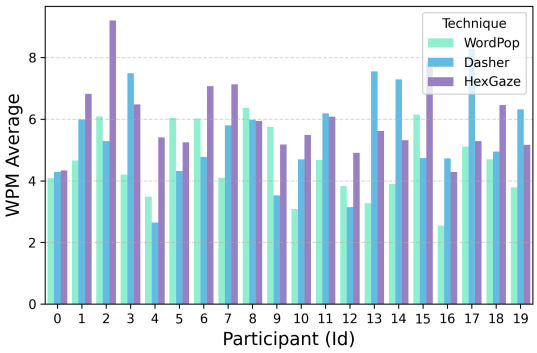
For the text entry rate, the results obtained can be seen in Figure 4.2. By looking at the boxplot and violinplot (Figure 4.2a and Figure 4.2b) we can see that WordPop demonstrated the lowest performance among the three techniques, with an average speed of 4.59 WPM. In contrast, Dasher and HexGaze achieved higher averages of 5.40 WPM and 5.96 WPM, respectively, with HexGaze showing the best typing performance overall. This trend is also reflected in the median values, where WordPop reached 4.26 WPM, while Dasher and HexGaze achieved 5.23 WPM and 5.78 WPM, respectively.



(a) BoxPlot for the WPM per technique.



(b) ViolinPlot for the WPM per technique.



(c) WPM BarPlot Comparison of WordPop, Dasher and HexGaze.

Figure 4.2: WPM Visualization: (a) BoxPlot, (b) LinePlot (c) BarPlot. Significant statistical difference between HexGaze and WordPop ($p < 0.0001$)

To further analyze these results, Figure 4.2c presents the average words per minute achieved by each participant for all three techniques. As shown, HexGaze consistently achieved the highest average WPM across most participants (50% of participants), followed by Dasher (35% of participants), while WordPop generally demonstrated lower performance (15% of participants). This pattern reinforces the previous statistical findings, indicating that HexGaze allowed for faster and more fluent text entry overall. We can also notice that results vary between participants, meaning each person performed better with some techniques than others. This suggests that individual comfort and familiarity with a technique can affect typing speed.

Dasher achieved the highest maximum value with 13.03 WPM, followed closely by HexGaze

with 11.58 WPM and WordPop with 10.56 WPM, making Dasher the technique that enabled the fastest typed sentence. However, when observing the lowest typing speeds, both WordPop and Dasher recorded very low minimum values of 0.79 WPM and 0.98 WPM, respectively, while HexGaze maintained a comparatively higher minimum of 2.93 WPM.

During the experiments, we noticed participants were having issues with Dasher and WordPop due to the way they were made. In the case of Dasher, correcting errors was very time consuming, and for participants who had not yet fully adapted to its mechanics, multiple mistakes within a sentence significantly increased typing time. For WordPop, the main source of delay came from the placement of the space key in the top right corner of the screen, which made selection difficult, particularly for participants wearing glasses, as slight gaze-tracking inaccuracies often resulted in extended attempts to trigger that key.

The Shapiro–Wilk test was used to check if the data followed a normal distribution. As shown in Table 4.1, all p-values are above the 0.05 threshold, indicating the data for WordPop ($p = 0.216$), Dasher ($p = 0.820$), and HexGaze ($p = 0.112$) failed to reject the Null Hypothesis and can be considered normally distributed. Given the repeated measures nature of the study (the same participants tested all techniques) and the assumption of normality being met, a Repeated Measures Analysis of Variance (RM-ANOVA) was used for the overall test. The test revealed a statistically significant difference between the techniques, with an $F(2, 38) = 6.79$ and a p-value of $p = 0.003$. Since $p < 0.05$, the Null Hypothesis that all techniques have the same WPM is rejected, indicating that the difference between at least two techniques is real. The effect size, η^2_G, was 0.164, a above average value, meaning that 16.4% of the variance in WPM is explained by the choice of technique.

Table 4.2: Post-Hoc Comparison Results (WPM).

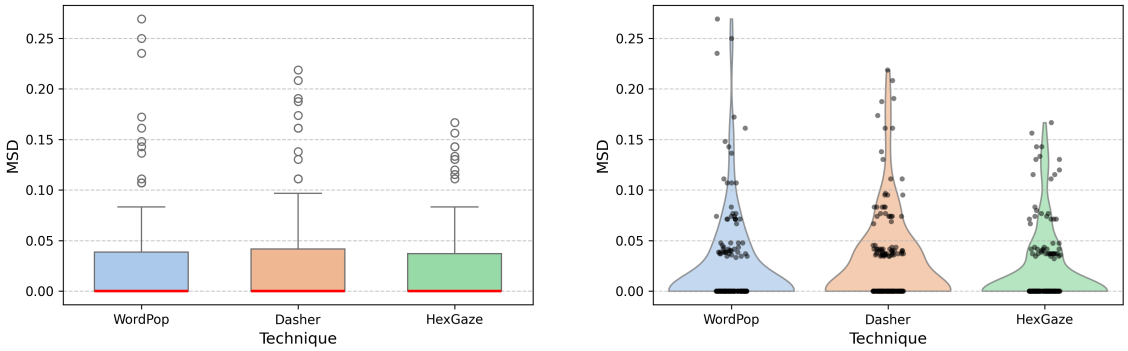
A	**B**	$\boldsymbol{T}$	$\boldsymbol{p}_{\text{corr}}$	Cohen's $\boldsymbol{d}$
Dasher	HexGaze	-1.391	0.5412	-0.412
Dasher	WordPop	1.850	0.2398	0.597
HexGaze	WordPop	5.392	0.0001	1.160

Since the RM-ANOVA test was significant, we conducted a t-test with Bonferroni correction post hoc comparisons to identify specific differences and the results are shown in Table 4.2. As we can see the difference between Dasher and HexGaze was not statistically significant ($p = 0.541$), nor was the difference between Dasher and WordPop ($p = 0.240$). However, there was a highly significant difference in WPM between HexGaze and WordPop since the corrected p-value (0.0001) is below the $\alpha = 0.05$ threshold. Given that Cohen's $d = 1.16$ and $t = 5.39$ are positive we can say there is a significant statistical difference where the mean WPM for HexGaze is higher than for WordPop, this confirms that HexGaze is significantly faster. In summary, WPM performance differs across techniques, with HexGaze being significantly superior to WordPop, while Dasher shows no significant difference from either.

Error Rate

For the MSD error rate, as shown in Figure 4.3, the results indicate that all three techniques produced very low error rates overall. WordPop and HexGaze achieved the lowest average MSD values, with 0.025 and 0.021, respectively, while Dasher showed a slightly higher mean error rate of 0.031. In terms of median values, all techniques achieved a score of 0.0, indicating that most trials were completed without any errors.

When observing the maximum recorded values, WordPop exhibited the highest error peak with 0.269, followed by Dasher with 0.219, and HexGaze with 0.167. This suggests that while errors were generally rare, when they did occur, WordPop tended to produce more severe deviations than the other techniques. This again may be due to the fact that some users could not reach the space key properly and while trying to select it they would select the button underneath by accident which would type a different word than the one intended and due to frustration they would not try to fix it.



(a) BoxPlot for the MSD per technique.

(b) ViolinPlot for the MSD per technique.

Figure 4.3: MSD visualization: (a) BoxPlot and (b) ViolinPlot. No significant statistical difference between techniques.

The Shapiro–Wilk test was used to verify the normality of the data. As shown in Table 4.1, WordPop met the normality assumption ($p = 0.269$), whereas Dasher ($p = 0.003$) and HexGaze ($p = 0.001$) did not. Since the normality assumption was violated for two of the techniques and given the repeated measures of the study, the non-parametric Friedman test was used instead of RM-ANOVA.

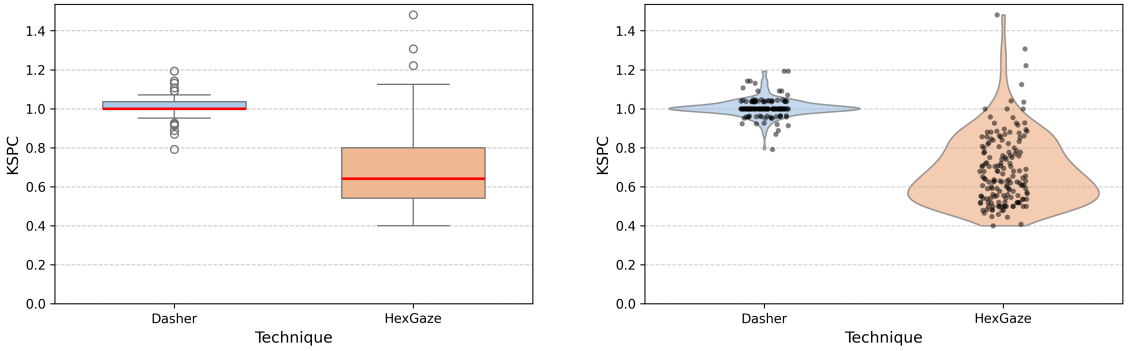
The Friedman test revealed no statistically significant differences in MSD error rate between the techniques, with $\chi^2(2) = 1.6$, $p = 0.449$, and a Kendall's $W = 0.04$, indicating a very small effect size. Given that $p > 0.05$, we fail to reject the Null Hypothesis suggesting that the error rates across WordPop, Dasher, and HexGaze were statistically equivalent.

In summary, all techniques achieved similarly low MSD error rates, with HexGaze showing the lowest average values and WordPop and Dasher falling close behind. However, the overall statistical analysis confirms that these differences are not significant.

Keystrokes Per Character

As mentioned earlier during data processing, the WordPop results for KSPC were removed. Upon inspection, it became clear that keystrokes were being counted incorrectly, producing unrealistic values that would distort the comparison. To ensure valid statistical analysis, only Dasher and HexGaze were included in this metric.

Regarding the KSPC (Keystrokes Per Character) metric, Figure 4.4 illustrates that HexGaze achieved the lowest values, with an average of 0.68, indicating higher input efficiency where users could type more words with less button presses (selected). In contrast, Dasher showed a higher average of 1.01, meaning participants needed more keystrokes to produce the same amount of text. Given that for Dasher we have to type every letter this is to be expected since you can not cut corners. The median values reflect a similar pattern, with HexGaze at 0.64 and Dasher at 1.00. When looking at the extremes, HexGaze recorded the lowest minimum of 0.40, while Dasher reached 0.79. The highest KSPC observed was 1.19 for Dasher and 1.48 for HexGaze, though overall, HexGaze maintained more consistent performance across users.



(a) BoxPlot for the KSPC per technique.

(b) ViolinPlot for the KSPC per technique.

Figure 4.4: KSPC visualization: (a) BoxPlot and (b) ViolinPlot. Significant statistical difference between HexGaze and Dasher ($p < 0.001$)

Because one technique was excluded, a different statistical approach was required. We began by testing the normality of the differences between Dasher and HexGaze using the Shapiro–Wilk test, which confirmed that the differences were normally distributed ($p = 0.4416$).

Table 4.3: T-test results for KSPC (Dasher vs. HexGaze).

Test	**T**	**df**	**p**	**Cohen's d**
Paired T-test	13.868	19	2.17×10^{-11}	4.324

This allowed the use of a paired-samples t-test. The results, seen in Table 4.3, indicated a clear and statistically significant difference between the two techniques ($t(19) = 13.87$, $p < 0.001$). Since $p < 0.05$, the result is considered statistically significant, confirming that the difference is not random. The Cohen's d value of 4.32, represents how large the difference is in standardized

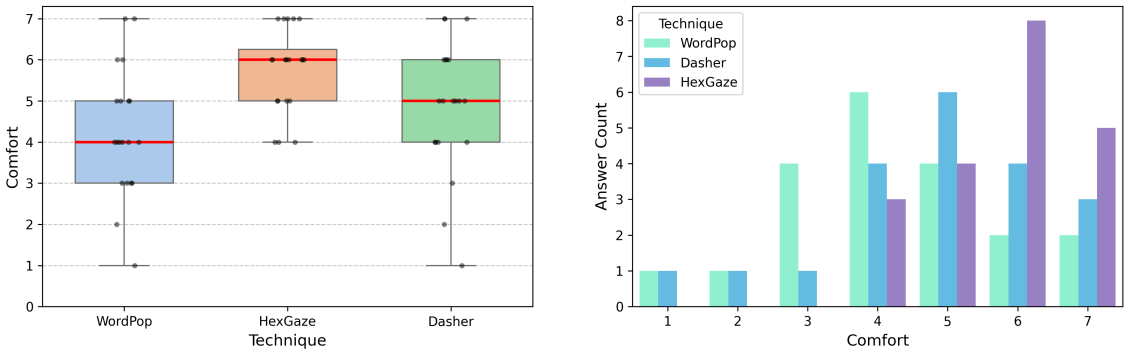
terms. In this case, $d = 4.32$ is extremely large, showing a substantial gap in performance between the two techniques. On average, Dasher ($M = 1.01$) required more keystrokes per character than HexGaze ($M = 0.68$).

In conclusion, the KSPC results show that HexGaze is considerably more efficient than Dasher, requiring fewer keystrokes to generate text and offering a much smoother typing experience.

Feedback

Participants feedback in terms of comfort, eye fatigue, perceived typing speed, ease of learning and ease of typing was acquired through a questionnaire between parts of the session and later analysed. These metrics were collected using a 7-point Likert scale of where 1 is very poor and 7 is excellent.

Relating to comfort the ratings for each technique are presented in Figure 4.5. Overall, HexGaze received the highest comfort ratings, with an average of 5.75 and a median of 6 on a 7-point scale (Figure4.5a). The lowest rating recorded for HexGaze was 4, indicating that most participants found the technique comfortable to use. Dasher also received generally high comfort ratings, with an average of 4.85 and a median of 5, though one participant reported a minimum score of 1. WordPop achieved the lowest comfort levels, with an average of 4.25, a median of 4, and a minimum rating of 1, suggesting a wider range of perceived comfort across users.



(a) BoxPlot for the perceived comfort per technique.

(b) BarPlot for the perceived comfort per technique.

Figure 4.5: Perceived comfort visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) BarPlot.

The lower comfort scores for WordPop align with the issues previously discussed. Participants frequently expressed frustration with the placement of the space key in the top right corner of the screen, which required larger gaze movements and often led to repeated selection attempts. Similarly, Dasher also caused some discomfort, particularly for participants still learning its navigation mechanics. Correcting sentences demanded frequent gaze and head movements to locate intended letters, which increased fatigue over time. In contrast, HexGazes interface offered more stable centralized target zones and smoother interaction, leading to consistently higher comfort ratings.

Because comfort ratings are ordinal data, a non-parametric Friedman test was applied to

compare the three techniques. The test revealed a statistically significant effect of technique ($\chi^2(2) = 12.79$, $p = 0.0016$), indicating that at least one technique differed from the others.

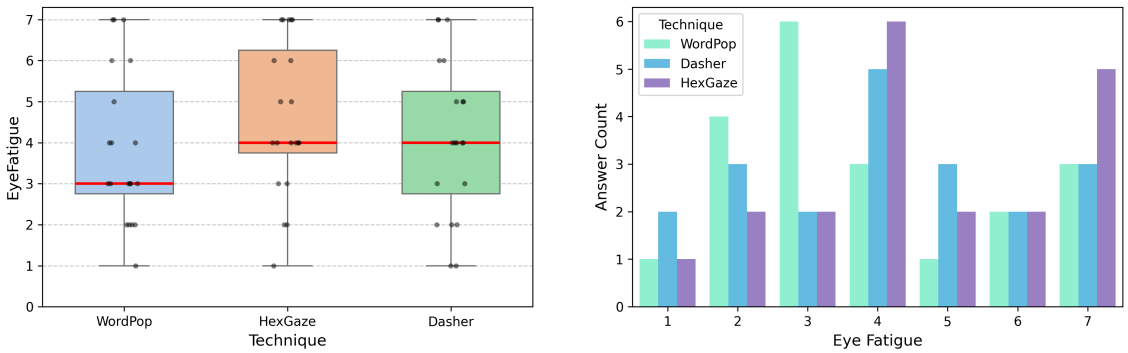
To identify where those differences occurred, post-hoc Wilcoxon tests with Bonferroni correction were conducted.

Table 4.4: Pairwise Wilcoxon comparisons with Bonferroni correction for Comfort ratings.

A	B	W-val	p-corr	Cohen's d
Dasher	HexGaze	26.0	0.129	-0.67
Dasher	WordPop	42.0	0.297	0.38
HexGaze	WordPop	0.0	0.006	1.14

The comparison test results can be seen in Table 4.4 and they show that between WordPop and HexGaze there is a significant difference ($p = 0.006$), showing that participants rated HexGaze as significantly more comfortable. Differences between WordPop and Dasher ($p = 0.297$) and between Dasher and HexGaze ($p = 0.129$) were not statistically significant, meaning there is not much of a difference between them.

Eye Fatigue, on the other hand, showed more similar values across techniques as seen in Figure 4.6. HexGaze had the highest comfort for the eyes, with an average rating of 4.6 and a median of 4. Dasher followed closely with an average of 4.1 and a median of 4, while WordPop showed slightly lower ratings, averaging 3.85 with a median of 3. The lower ratings for WordPop and Dasher may be explained by their bright white backgrounds, which participants mentioned as causing more eye strain during the tests.



(a) BoxPlot for the perceived eye fatigue per session.

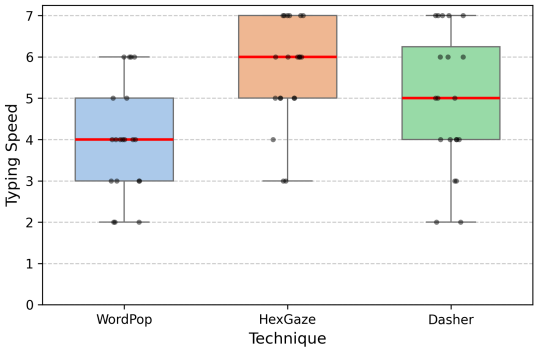
(b) BarPlot for the perceived eye fatigue per session.

Figure 4.6: Perceived eye fatigue visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) BarPlot.

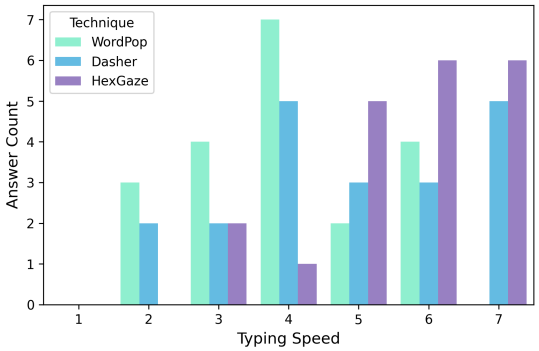
The Friedman test revealed no statistically significant effect of technique on Eye Fatigue, $\chi^2(2) = 3.49$, $p = 0.175$. This indicates that perceived eye strain did not differ significantly between the three methods.

Perceived Typing Speed ratings, shown in Figure 4.7, revealed clear differences among the techniques. In Figure 4.7a we can see that HexGaze achieved the highest ratings, with an average

of 5.65, a median of 6, and values ranging from 3 to 7.



(a) BoxPlot for the perceived typing speed per session.



(b) BarPlot for the perceived typing speed per session.

Figure 4.7: Perceived typing speed visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) BarPlot.

Dasher followed with an average of 4.9 and a median of 5, while WordPop had the lowest perceived speed, averaging 4 with a median of 4. The Friedman test indicated a statistically significant difference between techniques ($\chi^2(2) = 7.97, p = 0.018$).

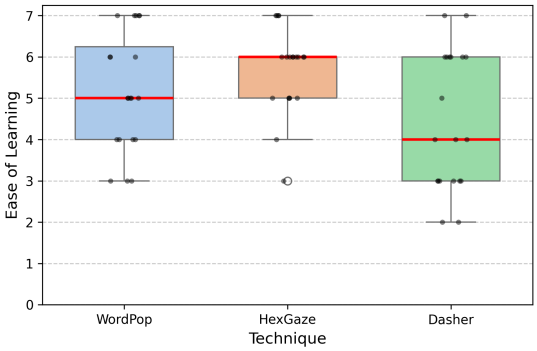
Table 4.5: Pairwise Wilcoxon Post Hoc Test Results for Perceived Typing Speed

A	B	W-val	p-corr	Cohen's d
Dasher	HexGaze	39.0	0.388	-0.50
Dasher	WordPop	37.0	0.184	0.59
HexGaze	WordPop	12.0	0.006	1.27

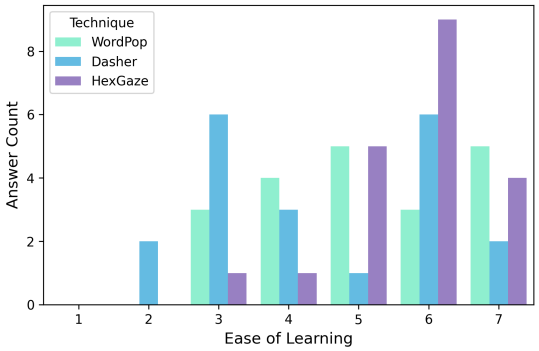
After making comparisons using the Wilcoxon test with Bonferroni correction we were able to identify significant difference between HexGaze and WordPop ($p = 0.006$), while no significant differences were observed between Dasher and WordPop ($p = 0.184$) or between Dasher and HexGaze ($p = 0.388$). These results indicate that participants perceived HexGaze and Dasher as the fastest techniques overall, while WordPop was seen as noticeably slower. Once again, this lower ratings can be explained by its usability issues, especially the extra head and eye movements required, which made the technique feel slower.

Interestingly, the perceived typing speeds were very similar to the actual results from the WPM metric, where HexGaze also had the highest speed, followed by Dasher and then WordPop. This shows that participants had a good sense of how fast they were typing with each technique, and their opinions matched the real performance data.

We also questioned participants about how easy it was to learn each technique and gathered the results shown on Figure 4.8.



(a) BoxPlot for the perceived ease of learning per session.



(b) BarPlot for the perceived ease of learning per session.

Figure 4.8: Perceived ease of learning visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) BarPlot.

As we can notice Dasher was the most difficult virtual keyboard to learn, with an average rating of 4.45, a median of 4 (Figure 4.8a), and a minimum score of 2. In contrast, WordPop and HexGaze were easier to learn, with WordPop achieving an average of 5.15 and a median of 5, while HexGaze showed the highest ease of learning, with an average rating of 5.7 and a median of 6.

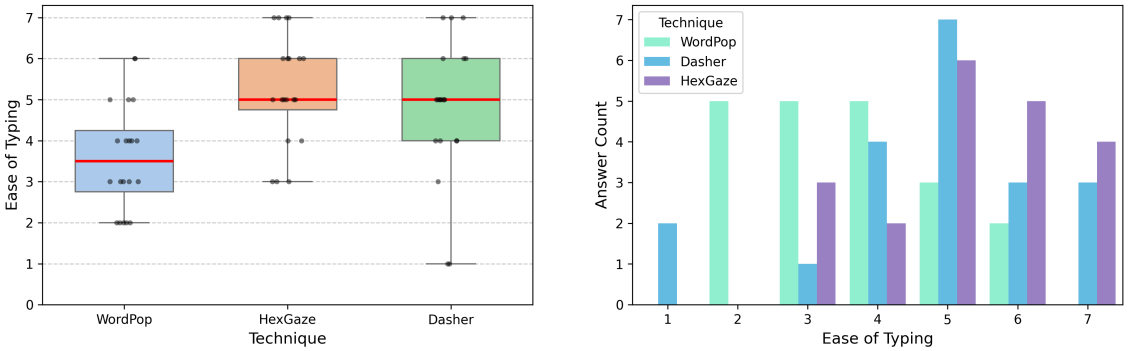
The Friedman test indicated a statistically significant difference between techniques ($\chi^2(2) = 7.40$, $p = 0.025$). Pairwise Wilcoxon comparisons with Bonferroni correction showed a significant difference between Dasher and HexGaze ($p = 0.009$) seen in Table 4.6, while no significant difference was found between WordPop and the other two techniques.

Table 4.6: Pairwise Wilcoxon Post Hoc Test Results for Ease of Learning

A	B	W-val	p-corr	Cohen's d
Dasher	HexGaze	8 .0	0.009	-0.90
Dasher	WordPop	48.0	0.541	-0.45
HexGaze	WordPop	54.0	0.508	0.44

These results suggest that participants found HexGaze the easiest technique to learn, followed closely by WordPop, while Dasher was perceived as the most difficult. This lower rating for Dasher likely comes from its unique input method and requires more time and practice to understand. In contrast, both HexGaze and WordPop offer a more familiar and intuitive experience, as they share a similar structure based on selecting groups of letters, making them easier and faster for new users to learn.

We then asked participants how easy it was to type using the virtual keyboards and the answers can be seen in Figure 4.9. These graphs show clear differences among the techniques and we can tell participants had different opinions about each technique no matter how they performed. WordPop received the lowest ratings, with an average of 3.6, a median of 3.5, and scores ranging from 2 to 6 (Figure 4.9a), suggesting users found it the most difficult to write with.



(a) BoxPlot for the perceived ease of typing per session.

(b) BarPlot for the perceived ease of typing per session.

Figure 4.9: Perceived ease of typing visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) BarPlot.

Dasher performed somewhat better, achieving an average of 4.75 and a median of 5, however for some users it was considered harder to type than WordPop given that it has a minimum value of 1. HexGaze achieved the highest scores overall, with an average of 5.25, a median of 5, and values between 3 and 7, indicating that users found it smoother and more intuitive to use.

The Friedman test confirmed that these differences were statistically significant ($\chi^2(2) = 8.03$, $p = 0.018$). Post hoc-Wilcoxon tests with Bonferroni correction (Table4.7) further showed significant differences between WordPop and HexGaze ($p = 0.012$) as well as WordPop and Dasher ($p = 0.042$), while no significant difference was found between HexGaze and Dasher ($p = 0.636$). In line with these findings, it appears that WordPop's complex interactions and frequent head movements made writing more difficult for some participants, whereas HexGaze and Dasher provided more fluid and efficient text entry experiences once participants became accustomed to their operation.

Table 4.7: Pairwise Wilcoxon Post Hoc Test Results for Ease of Typing

A	B	W-val	p-corr	Cohen's d
Dasher	HexGaze	38.0	0.636	-0.33
Dasher	WordPop	29.5	0.042	0.76
HexGaze	WordPop	15.5	0.012	1.25

Finally participants were asked at the end of the session which virtual keyboard was their favorite. Participants voted the most on HexGaze as can bee seen in Figure 4.10 with a total of 13 votes. Even though Dasher performed better than WordPop, when it came to the other metrics, participants still chose WordPop over Dasher as their favorite app with 5 votes for WordPop and 2 for Dasher. To analyse these categorical data a Chi-square test was used to check if there was a difference in the preferences. The test showed a significant difference ($\chi^2 = 9.700$, $p = 0.0078$), meaning that participants clearly favored one technique over the others. These results match the previous findings, showing that while HexGaze was the most liked overall, WordPop and Dasher

still had a few supporters.

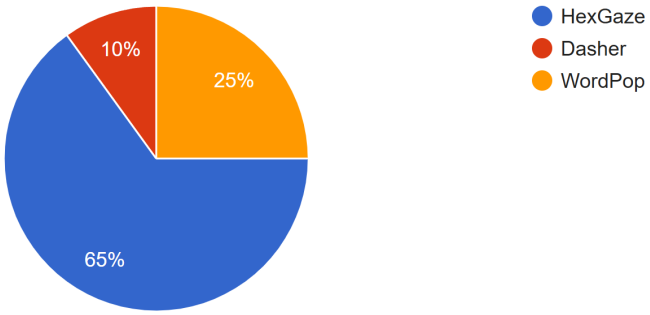


Figure 4.10: Pie Chart of Favorite App Comparison of WordPop, Dasher and HexGaze

Overall, the results provide a comprehensive view of how each text entry technique was perceived across multiple aspects of usability. HexGaze consistently achieved the highest ratings, being regarded as the most comfortable, least fatiguing, and easiest to learn and use, while also delivering the highest perceived and real typing speed. These findings align closely with its strong objective performance, suggesting that users not only performed better with HexGaze but also felt more at ease while doing so. Dasher, although showing moderate results overall, was limited by its unconventional control method, which made it harder for participants to master initially. However, once understood, users reported a reasonable level of comfort and speed. In contrast, WordPop tended to score lower in most categories, particularly in comfort and eye fatigue, mainly due to its bright background and higher demand for head and eye movements, which increased strain and reduced typing efficiency, however when understood most participants also performed quite well and achieved high typing performance scores. Taken together, these outcomes highlight HexGaze as the most balanced and user-friendly technique among the three, though it too could benefit from further refinement to improve comfort and reduce effort and eye fatigue over extended use given that it was where it performed worst.

4.2 Study II - Evolution Over Time

In this section, we analyse how users improved over time while using our technique. Unlike the first experiment, which compared HexGaze with other methods, this study focuses solely on user progression across multiple sessions. The goal was to evaluate how participants performance, comfort, and overall experience evolved with continued practice, helping us understand the learning curve and long-term usability of HexGaze.

4.2.1 Participants

We recruited 11 participants by word of mouth for this experiment, comprising seven males and four females, aged between 18 and 52 with an average of 23.8 (SD = 10.04), all without disabilities and three of which had prior experience using eye tracking (Less than 1 hour of experience).

Among them, two wore glasses or contact lenses, none were native English speakers, but they were all fluent in it. None of them participated in the first study.

4.2.2 Apparatus

The tests were conducted on a desktop computer equipped with a i9-14900K CPU at 3.2 GHz base speed and 32 GB of RAM, running Windows 11, and connected to a 28-inch monitor with a resolution of 3840 × 2160. Eye tracking was performed using a Tobii Eye Tracker 4C, mounted underneath the monitor with participants standing 60 cm in front of it. All testes where conducted in a quiet, well lit and calm place with only the presence of the investigator and the participant. All tests were in the same place.

4.2.3 Experimental Procedure

The study was conducted over eight typing sessions to evaluate participants progress over time. A total of 72 phrases were selected from the MacKenzie and Soukoreff dataset [24], providing a balanced mix of simple phrases, complex phrases, and sentences containing many pronouns and short words. These phrases were divided into nine groups of eight: one group used for training and eight groups used across the sessions. Each session was assigned a specific group of phrases, meaning that in session 1, all participants typed phrases from group 1, in session 2 from group 2, and so on, maintaining consistency throughout the experiment ensuring that each participant progress could be directly compared across sessions and following a within-subject design.

At the start of the experiment, we explained to each participant the purpose and objectives of our study, emphasizing that it was the technique being evaluated rather than their personal performance. Participants were then asked to sign a consent form and complete a short questionnaire about their age, vision, whether they used glasses or contact lenses, and have had any eye typing experience in the past.

Before each typing session began, the eye tracker was calibrated for the participant. The experiment only proceeded once we confirmed that the estimated gaze aligned closely with the participants actual gaze.

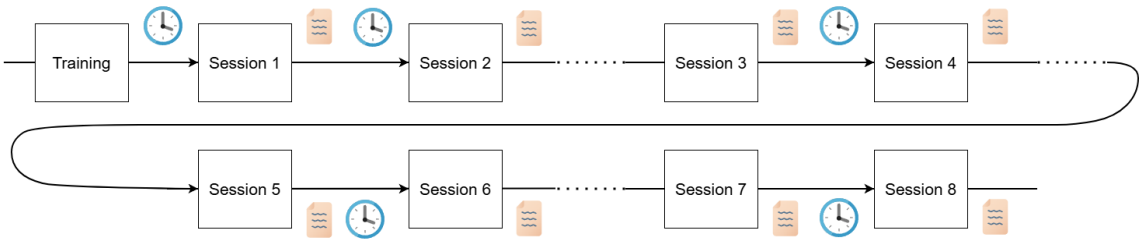


Figure 4.11: Steps of the experimental procedure for study II (The clock icon represents a 10-minute break, the paper icon indicates the questionnaire phase, and the dots signify the passing of days between sessions).

Although a small margin of error was inevitable, the calibration was generally precise. If participants reported inaccuracies, the eye tracker was recalibrated until the tracking felt accurate and comfortable for them.

Once calibration was complete, the typing sessions began. The overall procedure timeline is illustrated in Figure 4.11. In the first typing session, participants were given approximately ten minutes to practice typing with the technique before starting the actual test. During testing, participants were first shown the phrase to type and given time to memorize it. Although the phrase remained visible on screen, this approach encouraged a more natural typing process similar to when people already know what they intend to write without constantly rereading. However, the phrase stayed available in case they needed a quick reference. When ready, the experimenter pressed a key to start the typing phase. After completing each phrase, the same key was pressed again to finish the typing of the sentence and display the next one. This pause allows participants to rest and refocus on the new sentence before continuing. To ensure accurate data, the system recorded the timestamp of each typed letter and used the time of the last letter as the reference for calculating typing speed. This approach ensured that any delay in pressing the key after finishing the sentence did not affect the speed measurements. This cycle is then repeated until all eight phrases for the session were completed.

After typing all eight phrases, participants completed a short questionnaire evaluating the technique. The form included questions about comfort, eye fatigue, ease of typing, and perceived typing speed. All items were rated on a 7-point Likert scale, ranging from 1 (very poor) to 7 (excellent). After completing the questionnaire, participants were given a 10-minute break before proceeding to the next typing session.

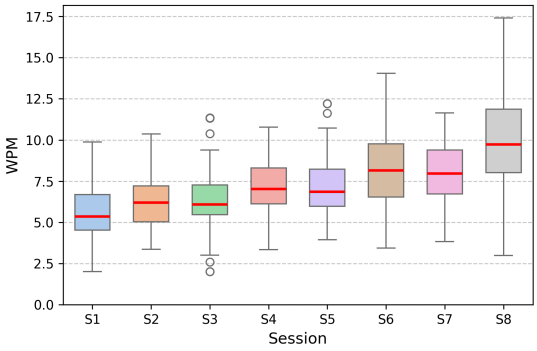
This procedure was repeated four times, with each participant typing eight phrases per session, totalling eight sessions overall. The sessions were conducted in pairs of two per day across four days, with no more than 48 hours between them to maintain consistency and minimize learning decay. At the end of the final session, participants were thanked for their time and contribution.

4.2.4 Experimental Results

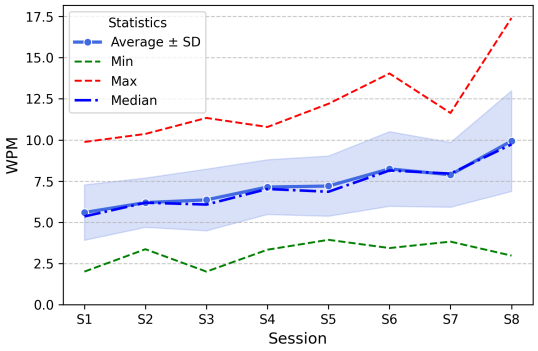
Once the experiment concluded, we processed and examined the collected data to assess participants' performance and experience. The analysis included the same measures used in Study I, typing speed (WPM), accuracy (MSD), efficiency (KSPC), comfort, and subjective user evaluations. We also used the same tests (Shapiro-Wilk) to assess the normality of the data. Then, based on the normality test results we used the appropriate statistical test to determine differences between sessions. This was also the case for feedback where the same methods were used to test for a difference between conditions (Friedman Test). Since one of the participants was not able to perform consecutive sessions with no more than 48 hours between them, their data was removed. Thus, in the end we have data from ten users.

Text Entry Rate

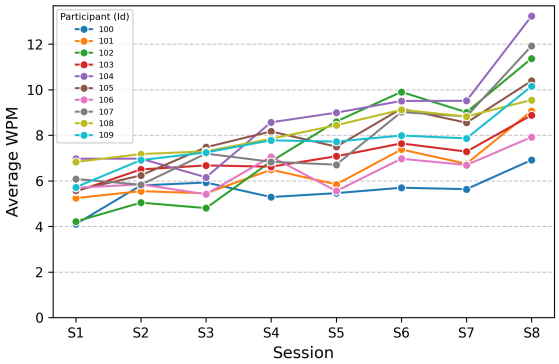
In Figure 4.12, the evolution of participants typing speed across sessions is shown. Figure 4.12c illustrates each participants individual progress, while Figure 4.12b presents the overall average performance across all participants, along with lines representing the maximum, minimum, and standard deviation values. The standard deviation line indicates how consistent the results were among participants where smaller deviations mean their performance became more similar over time. From these graphs, we can observe a clear improvement in typing speed across sessions, suggesting that participants became more proficient and stable with practice.



(a) BoxPlot for the WPM per session



(b) LinePlot for the WPM per session



(c) LinePlot for the average WPM per participant per session

Figure 4.12: WPM Visualization: (a) BoxPlot, (b) LinePlot (c) LinePlot per Participant.

As shown in the graphs, participants performance improved steadily across sessions, with the final session having the highest results as expected. The early sessions started with lower values, but overall, the data demonstrates a clear and consistent upward learning curve for the technique. Individual progress can also be observed, with most participants showing gradual increases in their typing speed over time. Table 4.8 provides detailed statistics for each session, including the maximum, minimum, average, and standard deviation values. The average WPM increased from 5.59 in the first session to 9.94 in the last, while the maximum WPM rose from 9.88 to 17.41, indicating a strong learning effect, with 50% of the participants reaching an average of above 10 WPM in

the last session and 80% above 8 WPM. In contrast, the minimum values fluctuated slightly from session to session, likely reflecting occasional difficulty with specific phrases. However, these remained relatively stable overall, suggesting that even slower participants were able to maintain consistent performance as they progressed.

Table 4.8: WPM Statistics per Session.

Metrics	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
Average	5.59	6.18	6.36	7.15	7.19	8.24	7.90	9.94
Median	5.35	6.2	6.08	7.02	6.85	8.15	7.98	9.71
Min	2.01	3.37	2.01	3.34	3.94	3.44	3.83	2.98
Max	9.88	10.37	11.34	10.79	12.20	14.04	11.64	17.41
Standard Deviation	1.60	1.42	1.77	1.60	1.75	2.16	1.92	2.91

The standard deviation values shown in Table 4.8 also reveal an interesting pattern. In the early sessions, the deviation was relatively smaller, suggesting that participants performances were closer together since everyone was still learning the technique. As the sessions progressed, the deviation increased, indicating that individual differences became higher. Some participants adapted quickly and aimed to reach higher speeds in the end, while others progressed more gradually, maintaining a pace they found comfortable.

Overall, the results show a clear upward trend in WPM across sessions, demonstrating consistent improvement and growing familiarity with the technique, even if the rate of progress varied between individuals.

Additionally, we observed an interesting behaviour during the experiment, where participants tended to type more slowly at the first session of each day but became faster at the second session of the days, likely due to regaining confidence as they re-familiarized themselves with the technique and remembered the tricks they learned. They also made improvements each day with every day having a last session with bigger performance than the first.

This trend is visible in Figure 4.12a, where the boxplot of each even-numbered session (the second session of the day) closely matches the boxplot of the following odd-numbered session (e.g., Session 2 and Session 3). It is also supported by the data in Table 4.8, which shows small increases in average WPM between the first and second sessions of each day (typically around 0.7 to 1 WPM) while the transition from the second session of one day to the first of the next shows little change, confirming that performance resets slightly after rest periods but quickly recovers.

When we compare the two sessions done on the same day (e.g., Session 1 vs. Session 2), a consistent pattern emerges where performance is always higher in the second session. This suggests that the first session of each day works like a warm-up, where participants remember what they learned before and learn more, and the last session shows the improvement they made throughout the day.

The boxplot also illustrates the increase in performance variability over time mentioned be-

fore. Early sessions (S1–S3) display tighter boxes, indicating that participants typing speeds were relatively close, while later sessions (S6–S8) show a wider spread, suggesting greater individual differences.

We used the Shapiro-Wilk test to analyse the normality of the data and it confirmed that the data followed a normal distribution across all sessions ($p > 0.29$ for all cases), indicating that parametric tests could be applied safely since the data met the normality assumption ($p > 0.05$). The W values ranged from 0.912 to 0.991, suggesting no major deviations from normality.

A one-way repeated measures ANOVA revealed a significant effect of session on performance, $F(7, 63) = 28.35$, $p < .001$, $\eta_g^2 = 0.56$, indicating that participants typing speeds differed significantly across sessions.

Table 4.9: Post-hoc t-test comparisons with Bonferroni correction and effect size (Cohen's d). Only significant differences are presented.

Comparison	**Direction**	p_{corr}	**Cohen's** d
S1 – S4	S1 < S4	0.0011	-1.64
S1 – S6	S1 < S6	0.0025	-2.30
S1 – S7	S1 < S7	0.0022	-2.07
S1 – S8	S1 < S8	0.0005	-2.91
S2 – S6	S2 < S6	0.0335	-1.94
S2 – S8	S2 < S8	0.0042	-2.63
S3 – S8	S3 < S8	0.0078	-2.39
S4 – S8	S4 < S8	0.0051	-1.86
S5 – S6	S5 < S6	0.0282	-0.80
S5 – S8	S5 < S8	0.0019	-1.70
S6 – S8	S6 < S8	0.0109	-1.04
S7 – S8	S7 < S8	0.0015	-1.27

Post-hoc pairwise t-test comparisons with Bonferroni correction revealed several significant differences between sessions as shown in Table 4.9, which contains only the sessions with statistical differences. We can see that the performance in Session 1 was significantly lower than in Session 4 ($p = .001$, $d = -1.64$), Session 6 ($p = .003$, $d = -2.30$), Session 7 ($p = .002$, $d = -2.07$), and Session 8 ($p < .001$, $d = -2.91$). Session 2 also showed significantly lower performance than Session 6 ($p = .034$, $d = -1.94$) and Session 8 ($p = .004$, $d = -2.63$). Similarly, Session 3 differed significantly from Session 8 ($p = .008$, $d = -2.39$), and Session 4 was also lower than Session 8 ($p = .005$, $d = -1.86$). Finally, Session 5 showed lower scores compared to Session 6 ($p = .028$, $d = -0.80$) and Session 8 ($p = .002$, $d = -1.70$), while Sessions 6 and 7 also differed from Session 8 ($p = .011$, $d = -1.04$ and $p = .002$, $d = -1.27$).

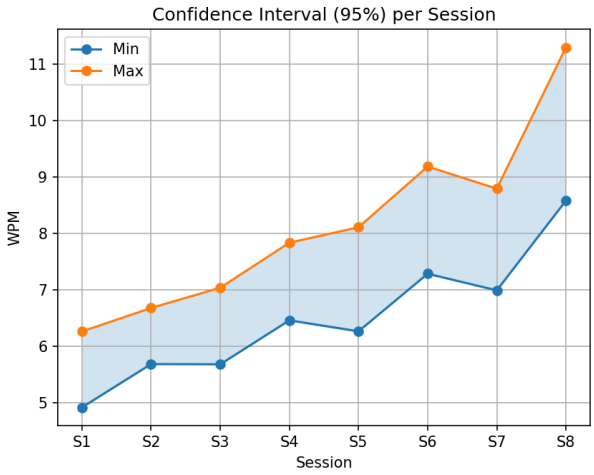


Figure 4.13: Confidence Interval for all sessions.

Overall, we can see a statistical difference where lower sessions have lower performance than higher sessions, and that Session 8 was significantly higher than all previous sessions, confirming a strong learning effect throughout the experiment. We can also notice that participants took around 3 Sessions to learn the technique, since the first significant difference in performance from Session 1 is Session 4.

We calculated the confidence interval for all sessions, and the results can be seen in Figure 4.13, which represent the range within which the true mean WPM for each session is expected to lie with 95% confidence.

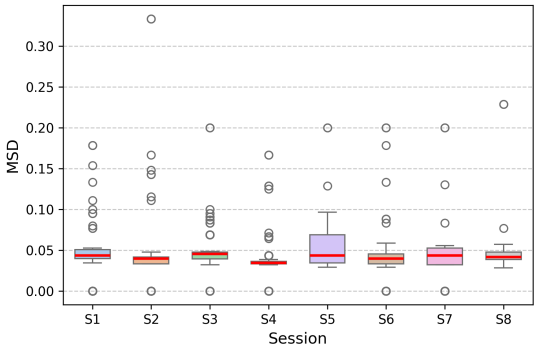
During the final sessions of the tests, we noticed that most users still were not using the technique to its full potential. Although a few more sessions could allow them to achieve an even better performance, with this results we were able to understand the learning curve of our virtual keyboards.

Error Rate

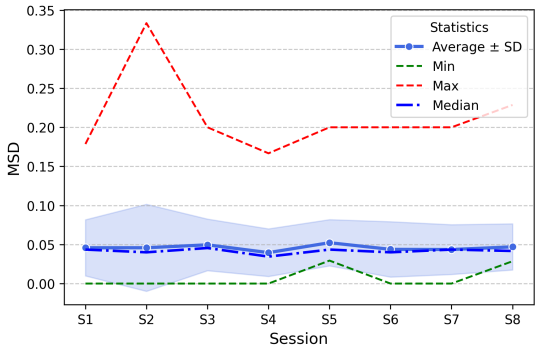
In the case of Error Rate our results varied slightly per session with some sessions having higher error rates while others have lower.

As shown in Figure 4.14, the MSD values remained consistently low across sessions, with only minor fluctuations. The average error rate hovered around 0.04–0.05, suggesting that participants maintained good accuracy throughout the experiment.

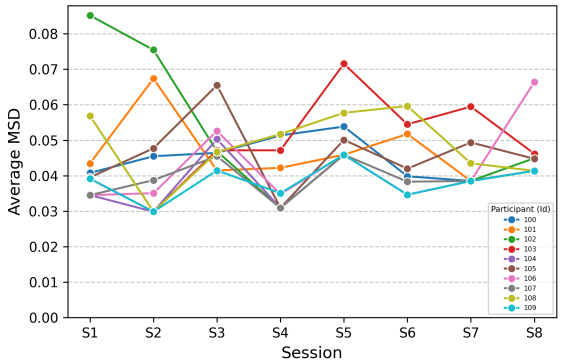
A slight increase in variability (SD area) can be observed during the initial sessions (S1–S3) in Figure4.14b, when participants were still adapting to the technique. This indicates that while most maintained stable accuracy, a few produced higher error values, increasing overall variation. As the sessions progressed, this variability gradually decreased and stabilized, suggesting that participants became more consistent in their performance.



(a) BoxPlot for the MSD per session



(b) LinePlot for the MSD per session



(c) LinePlot for the average MSD per participant per session

Figure 4.14: MSD Visualization: (a) BoxPlot, (b) LinePlot (c) LinePlot per Participant.

In Figure 4.14b and Figure 4.14c, these changes are evident where the highest error rates occurred in the early sessions. By the final sessions, the variability was minimal, and the standard deviation narrower, meaning participants were more likely to make similar types of mistakes.

The minimum MSD values remained near zero throughout the experiment, meanwhile the maximum values peaked in Session 2, likely due to early adaptation challenges, then dropped and stabilized after Session 4. A minor fluctuation in Sessions 3 and 5 may reflect the influence of sentence complexity, especially considering that none of the participants were native English speakers.

After testing the normality with the Shapiro-Wilk test the results indicated that the data were not normally distributed in any of the sessions ($p < 0.05$ for all), so the assumption of normality was not met. Because of that, we used a non-parametric test instead. The Friedman test showed a significant difference between sessions ($\chi^2 = 20.07$, $df = 7$, $p = 0.005$, $W = 0.29$). Therefore, we rejected the null hypothesis, indicating that the participants performance changed significantly across sessions.

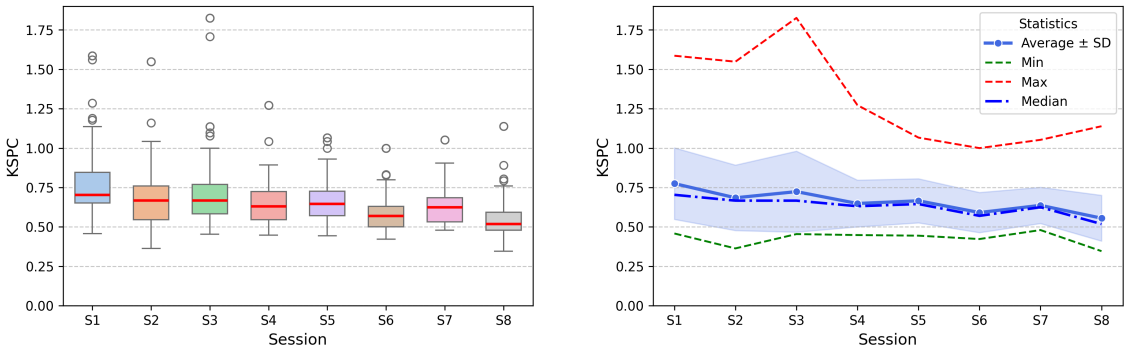
However, the post hoc Wilcoxon test with Bonferroni correction indicated that none of the pairwise comparisons reached statistical significance ($p > 0.05$ for all pairs), meaning that no two sessions differed significantly from one another according to this test.

Because the Bonferroni correction is quite conservative and can sometimes overlook smaller yet meaningful effects, we applied a more sensitive post hoc test which was the Nemenyi test, to further explore potential differences. The Nemenyi results revealed one significant pairwise difference between sessions: Session 4 and Session 5 ($p = 0.0122$), where Session 5 showed higher error rates than Session 4. This suggests that while overall performance differences across sessions were subtle, there was a localized increase in error rate between these two consecutive sessions.

Overall, the MSD progression suggests that accuracy remained stable even as participants typing speed improved (as seen in the WPM results), not having a big improvement throughout the sessions. This indicates that even though the mistakes were relatively the same as when they started, users still were able to type faster while committing around the same amount of mistakes.

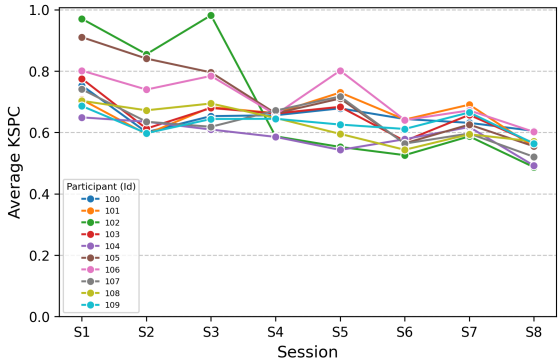
Key Strokes Per Character

The KSPC results show a clear downward trend across the eight sessions as can be seen in Figure 4.15.



(a) BoxPlot for the KSPC per session

(b) LinePlot for the KSPC per session



(c) LinePlot for the average KSPC per participant per session

Figure 4.15: KSPC Visualization: (a) BoxPlot, (b) LinePlot (c) LinePlot per Participant.

This indicates that participants progressively required fewer keystrokes per character as they became more familiar with the technique. The average KSPC, shown in Table 4.10, started at

approximately 0.77 in Session 1 and steadily decreased to 0.55 by Session 8, showing a consistent improvement in input efficiency with users typing almost two characters on average per key stroke. The median values follow the same pattern, declining from 0.70 in the first session to 0.52 in the last, reinforcing that this improvement was not limited to isolated participants but reflected a general shift across the group.

The minimum KSPC values remained relatively stable throughout the experiment with values ranging from 0.48 to 0.35, while the maximum values showed a sharp early peak in earlier sessions especially session 3 (1.83 max) before dropping and stabilizing to lower values. This was expected since in the early sessions, participants were still learning.

As training progressed, both the maximum values and the overall spread between minimum and maximum decreased as well as the standard deviation, which started higher in the earlier sessions (0.250) but got reduced later (0.146), reflecting a reduction in variability and a more efficient behaviour by the participants. By the end 80% of participants had a below 0.6 KSPC on average with the remaining 20% almost crossing that line (Figure 4.15c).

Table 4.10: KSPC Statistics per Session.

Metrics	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
Average	0.770	0.678	0.714	0.644	0.664	0.589	0.634	0.551
Median	0.704	0.667	0.667	0.629	0.645	0.569	0.625	0.519
Min	0.458	0.364	0.455	0.448	0.444	0.423	0.480	0.346
Max	1.586	1.548	1.826	1.273	1.067	1.000	1.053	1.139
Standard Deviation	0.250	0.227	0.269	0.170	0.165	0.140	0.144	0.146

The behaviours observed in the WPM analysis also appear in the KSPC metric. Although KSPC measures efficiency rather than speed, participants showed the same daily pattern of improvement. In general, lower KSPC values indicate fewer extra keystrokes and more efficient typing which also translates into speed. This measure, like WPM, consistently decreased across sessions within the same day. We can also see the same daily behaviour where participants performed roughly the same in the last session of a day and the first session of the next day.

These trends are visible in Figure 4.15a, where the boxplots of each even session (the last session of the day) align closely with those of the following odd session (e.g., Session 2 and Session 3). This pattern likely occurs because the first session acts as a re-familiarization period, during which participants recall the technique and regain confidence. By the end of the day, they typed with fewer corrections and unnecessary keystrokes.

When comparing the two sessions within each day (e.g., Session 1 and Session 2), we consistently observe lower KSPC values in the second session. This suggests that the first session functions as a warm-up, while the last session reflects the improvement achieved throughout that day.

The data in Table 4.10 also supports these observations. For example, within the first day, the

KSPC average decreases from 0.770 in the first session to 0.667 in the second, and it repeats for all the other days. Also, the change between the last session of one day and the first session of the next is minimal, for example both the last session of the first day and the first session of the next have a median of 0.667.

The boxplots also show how variability changed over time. Early sessions (S1–S3) display a wider spread, meaning that participants differed more in the number of extra keystrokes they made while learning the method. Later sessions (S6–S8) show a narrower range, indicating that participants became more consistent and efficient as the technique stabilized for everyone.

Table 4.11: Post-hoc comparisons (Nemenyi) for KSPC. Only significant differences are shown.

Comparison	**Direction**	p
S1 – S2	S1 > S2	0.0402
S1 – S6	S1 > S6	< 0.001
S1 – S7	S1 > S7	0.0304
S1 – S8	S1 > S8	< 0.001
S3 – S8	S3 > S8	0.0010
S4 – S8	S4 > S8	0.0227
S5 – S8	S5 > S8	0.0032

We used the Shapiro–Wilk test to evaluate the normality of the KSPC data across sessions. Results showed mixed normality where four sessions met the normality assumption ($p > 0.05$), while the remaining four did not. The W values ranged from 0.728 to 0.954. Since normality was not consistently met, we used non-parametric statistical tests for the KSPC analysis.

A Friedman test was conducted to assess whether KSPC values differed across the eight sessions. The test revealed a significant effect of session on performance, $\chi^2 = 47.63$, $p = 4.20 \times 10^{-8}$, $W = 0.68$, indicating strong differences in typing efficiency across sessions.

To perform the post-hoc pairwise comparisons, we used the Nemenyi test due to its higher sensitivity. It identified several significant differences between sessions as we can see in Table 4.11. Results show that Session 1 had significantly higher KSPC (worse efficiency) than Sessions 2, 6, 7, and 8. This indicates that participants made more unnecessary keystrokes at the beginning of the experiment, progressively improving as they became familiar with the technique.

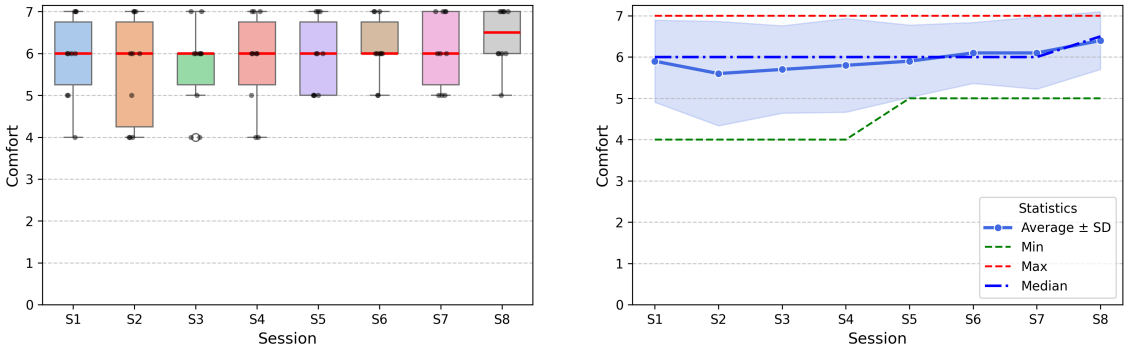
Sessions 3, 4 and 5 also showed significantly higher KSPC compared to Session 8 ($p = .001$, $p = .023$ and $p = .003$ respectively) confirming that efficiency continued to improve as participants gained experience but stabilizing later on.

Similar to the WPM results, the overall trend shows that higher-numbered sessions performed better than lower-numbered ones. Session 8 had significantly lower KSPC than almost all earlier sessions, demonstrating a strong learning effect throughout the study. We also observe that noticeable improvements occur after several sessions, participants required approximately three sessions before achieving clearly better efficiency, as the first significant difference relative to Session 1 emerges at Session 2 and becomes more pronounced from Session 6 onward.

Overall, the analysis indicates that participants became more consistent and efficient with each session, demonstrating clear learning effects where KSPC got reduced every day and improved mastery of the technique, with most being able to type on average two letters per keystroke.

Feedback

As mentioned before, we collected user feedback after every typing session regarding comfort, perceived typing speed, eye fatigue, and ease of writing, and then analysed the results.



(a) BoxPlot for the perceived comfort per session.

(b) LinePlot for the perceived comfort per session.

Figure 4.16: Perceived comfort visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) LinePlot.

Looking at the progression of the comfort ratings shown in Figure 4.16, several patterns become clear. In the boxplot (Figure4.16a), the early sessions show lower comfort levels and a wider spread of values, indicating greater variability in how participants felt at the beginning.

As the sessions progress, the comfort ratings gradually increase, and the variability decreases. This suggests that participants generally felt more comfortable and more consistent in their feedback during the later sessions.

We can also observe that the median comfort rating stayed constant at 6 for almost all sessions, increasing slightly to 6.5 in the final session. In both figures, the minimum comfort values rise from 4 in the early sessions (S1–S4) to 5 from Session 5 onward. This means that even the least comfortable participants reported higher comfort levels in the second half of the study, suggesting a reduction in very low comfort scores as they became more familiar with the technique.

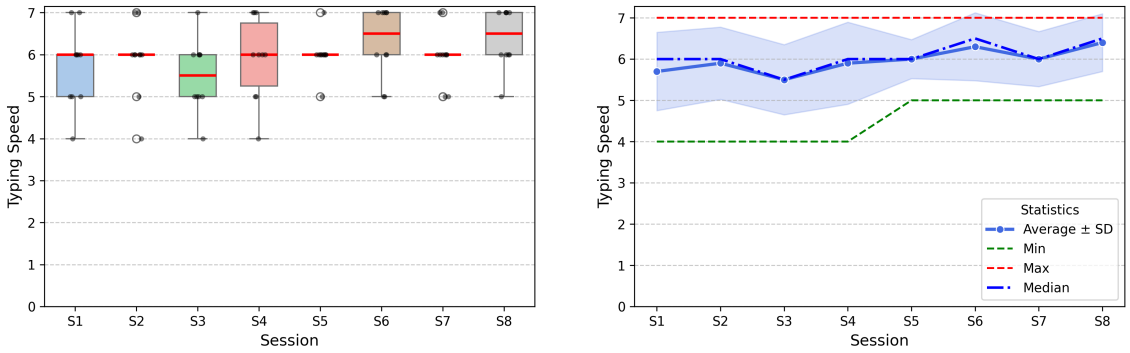
In Figure 4.16b, the mean comfort rating shows a steady upward trend, increasing from 5.9 in the first session to about 6.4 in the final session. This improvement indicates that, on average, participants felt progressively more comfortable as they gained experience. The statistical analysis of the comfort ratings revealed a significant overall effect across sessions. Since data is Ordinal we can skip the normality test and apply the non-Parametric Friedman test. The results indicated that comfort changed over time, with $\chi^2(7) = 14.697$, $p = 0.040$. To identify where these differences occurred, pairwise Wilcoxon tests with Bonferroni correction were performed.

Almost all comparisons resulted in non significant p-values ($p > 0.05$), suggesting that most differences were small. However, one comparison reached statistical significance which was Ses-

sion 3 and Session 8 ($p_{\text{corr}} = 0.031$) where Session 3 is smaller than Session 8, with a large effect size ($d = 0.88$). This result indicates that participants reported significantly higher comfort in the final session compared to Session 3.

Although the remaining comparisons were not statistically significant, several showed decreasing p-values toward later sessions (e.g., S2-S7 and S2-S8 both with p = 0.0625), consistent with the gradual upward trend observed in the results.

Overall, the statistical tests support the interpretation that comfort increased throughout the study, with the clearest improvement emerging between early and late sessions.



(a) BoxPlot for the perceived typing speed per session.

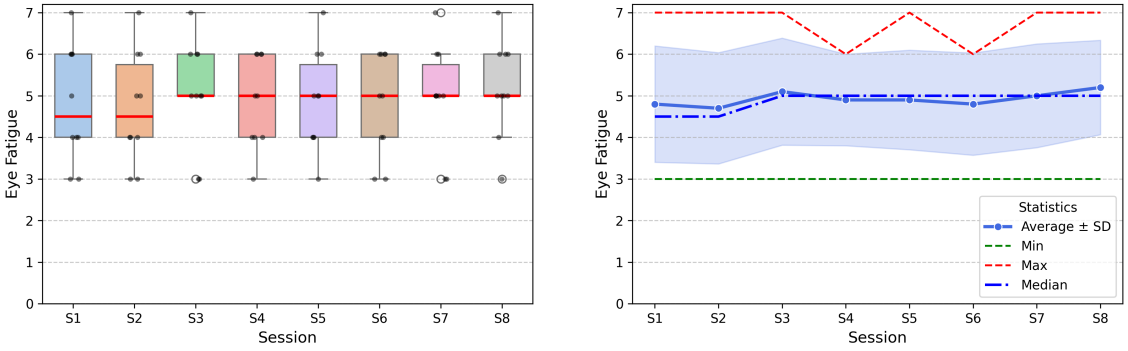
(b) LinePlot for the perceived typing speed per session.

Figure 4.17: Perceived typing speed visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) LinePlot.

Looking at the perceived speed ratings shown in Figure 4.17, a general upward progression can be observed across sessions. In the boxplot, the earlier sessions display slightly lower scores and more variability, reflecting differences in how quickly participants initially felt they were typing. As the study advances, the values tend to rise and become more consistent, suggesting that participants increasingly felt quicker and more confident with the technique. The median remains stable at 6 for most sessions, with small fluctuations down in Session 3 and upwards in Sessions 6 and 8. A similar pattern to comfort can be observed in the minimum scores where in the first four sessions (S1–S4), the lowest perceived speed reported by any participant is around 4, while from Session 5 onward the minimum increases to 5. This shift indicates that even participants who felt the slowest early on perceived themselves as faster and more comfortable with the technique in later sessions.

In the line plot, the mean perceived speed also shows a steady rise. It starts at 5.7 in Session 1, dips slightly in Session 3 (5.5), and then increases consistently toward the final session, reaching 6.4 in Session 8. This pattern suggests that participants gradually felt they were typing faster as they gained familiarity and control. The statistical analysis supports this progression. The Friedman test detected a significant effect across sessions, $\chi^2(7) = 16.534$, $p = 0.021$, indicating that perceived speed changed reliably over time. Pairwise Wilcoxon tests with Bonferroni correction showed that most differences were not statistically significant ($p > 0.05$), which is consistent with

the relatively smooth and gradual nature of the progression. However, two comparisons did reach significance which were S3-S6 and S3-S8, both with $p_{\text{corr}} = 0.031$, $d = 0.88$. In both cases, Session 3 shows lower perceived speed than the later sessions. These results confirm that participants felt noticeably faster in the middle and final stages of the study compared to earlier sessions. Also in WPM metric participants had a slight descent in performance with S7 being little slower than S6. This same performance can be seen here in the perceived speed where S7 is also slightly lower than S6 showing participants had good awareness of how fast they were typing. Although the remaining comparisons were not significant, several of them show decreasing p-values for later sessions (for example S1–S6 with $p = 0.109$ and S1–S8 with $p = 0.125$).



(a) BoxPlot for the perceived eye fatigue per session.

(b) LinePlot for the perceived eye fatigue per session.

Figure 4.18: Perceived eye fatigue visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) LinePlot.

Overall, the statistical results align well with the visual trends where perceived speed increased gradually throughout the study, with the clearest improvements observed when comparing early sessions to the middle and final ones.

The evolution of eye fatigue across the eight sessions is presented in Figure 4.18. Overall, the results reveal a mostly stable pattern, with only minor fluctuations in how tired participants felt throughout the study. For simplicity, higher values indicate better results, meaning that the higher the score, the less eye fatigue the user experienced.

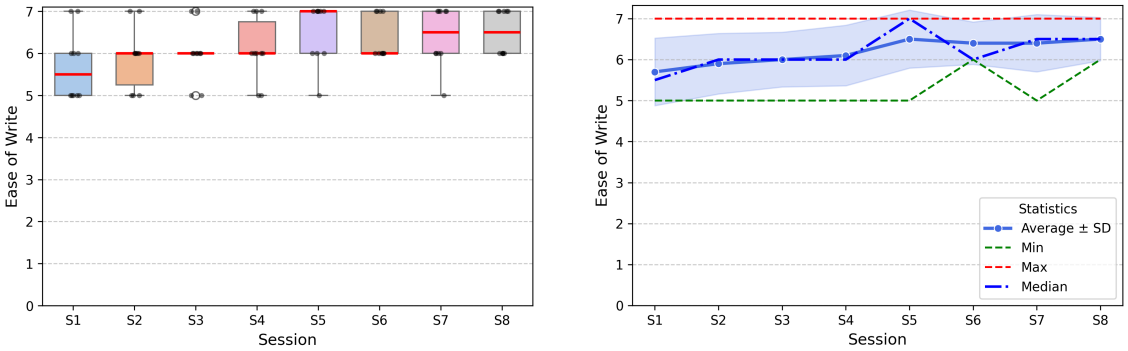
In the boxplot (Figure 4.18a), the general distribution remains fairly consistent across sessions. The minimum value stays fixed at 3 for all sessions, indicating that even the participants reporting the least fatigue remained at the same level from start to finish. The medians for Sessions 1 and 2 are slightly lower (4.5), after which all remaining sessions converge at a median of 5. This shift suggests that participants' perceived eye fatigue stabilized early and remained unchanged for the rest of the experiment. The maximum values of the distribution also shows little movement where the maximum is 7 in most sessions, dropping only slightly to 6 in Sessions 4 and 6.

The line plot (Figure 4.18b) reinforces this interpretation. The mean eye fatigue values show only minimal variation across the study, starting at 4.8 in Session 1, dipping slightly in Session 2 (4.7), and gradually rising to 5.2 by Session 8. Despite these small changes, the progression does

not form a clear increasing or decreasing pattern. Instead, the averages fluctuate within a very narrow range, indicating that the typing technique had no meaningful positive or negative effect on visual strain throughout the sessions. However since its values revolve around 5 it is still a good overall result with good comfort ratings being achieved.

The statistical analysis supports this conclusion. The Friedman test did not detect a significant effect of session on eye fatigue, $\chi^2(7) = 5.05$, $p = 0.65$, with a small effect size ($W = 0.07$). This helps us verify that the small variations seen in the descriptive statistics are not systematic.

In summary, eye fatigue remained low and stable across all sessions. Participants did not show signs of accumulating visual strain over time, nor did they report meaningful relief as they became more familiar with the technique, remaining constant at an overall good fatigue value.



(a) BoxPlot for the perceived ease of writing per session.

(b) LinePlot for the perceived ease of writing per session.

Figure 4.19: Perceived ease of write visualization (Values in a 7-point Likert scale): (a) BoxPlot and (b) LinePlot.

Finally we also asked participants how easy it felt to type with the technique. The evolution throughout the eight sessions is shown in Figure 4.19. In contrast to the stable pattern found for eye fatigue, the ease of writing metric reveals a more noticeable upward trajectory across the study.

The boxplot (Figure 4.19a) shows a clear improvement over time where in the early sessions, the scores are generally lower, but around the midpoint users begin to feel more comfortable with the technique, leading to higher ratings. The median follows the same pattern, starting at 5.5, increasing to 6 in Sessions 2 and 3, remaining around 6 for the following sessions, and reaching 6.5 by the final session. Although there are small fluctuations between Sessions 4 and 6, the overall trend indicates a steady and gradual increase in perceived ease.

The line plot (Figure 4.19b) further highlights this progression. The mean values rise steadily from 5.7 in the first session to 6.5 in the last one. The increase is particularly notable up to Session 5, after which the curve levels off, indicating that participants reached good level of comfort with the technique and stabilization.

The statistical analysis confirms that these changes are not random. The Friedman test identified a significant overall effect of session, with $\chi^2(7) = 19.68$ and $p = 0.006$, indicating that ease of writing changed meaningfully across sessions. Then we applied the pairwise Wilcoxon tests

with Bonferroni correction that revealed two significant differences. Session 1 was significantly lower than both Session 5 and Session 6 (for both, $p_{\text{corr}} = 0.031$, $d = 0.88$). These comparisons highlight an improvement between the beginning of the study and the midpoint, consistent with the upward trend observed in the graphs shown.

Although most other comparisons did not reach statistical significance, many showed decreasing p-values from early to later sessions, reinforcing the interpretation that ease of writing improved steadily as participants became more familiar with the system.

In summary, the results indicate a clear and meaningful increase in ease of writing over time, with participants experiencing noticeable gains in the first half of the study and maintaining that improved level afterwards.

4.3 Discussion

The results from the first experimental evaluation indicate that HexGaze outperformed both Dasher and WordPop across all major performance and usability metrics. HexGaze achieved the highest average WPM (5.96), surpassing Dasher (5.40) and WordPop (4.59). This improvement is notable given that users were interacting with all three techniques for the first time, suggesting that HexGaze offers a more intuitive and immediately accessible method of gaze-based text entry.

Error rate analysis reinforces this advantage. HexGaze produced the lowest MSD error rate (0.021), whereas Dasher and WordPop showed higher rates (0.031 and 0.025 respectively). Two main factors explain this: Dasher has a steeper learning curve and continuous zoom navigation that demands sustained attention, increasing the likelihood of mistakes. WordPop's poorly positioned space key led to frequent involuntary selections, inflating its error rates despite some appealing aspects of the technique.

Regarding keystroke efficiency, only Dasher and HexGaze could be evaluated because the WordPop values were invalid. As expected, Dasher produced more keystrokes per character (1.01) due to its continuous input design, while HexGaze required fewer key selections (0.68), indicating more efficient text entry. Although Dasher includes word prediction, it does not substantially reduce the number of selections for users. Instead, the KSPC results reflect how effectively participants can minimize inputs while typing. In this respect, HexGaze performed better, showing that beginners were able to type more efficiently with fewer keystrokes. Even though we can not compare it with WordPop which reduces key selection, we can still understand its overall performance for beginners and reinforcing HexGaze as a faster, more intuitive, and efficient technique.

User feedback results further strengthen what was mentioned. Participants consistently reported feeling more comfortable and less fatigued using HexGaze, which was also rated the easiest to learn and most preferred. WordPop ranked second, while Dasher was considered the most difficult. Some participants noted that Dasher felt smoother once they became familiar with it.

Overall, the first experiment demonstrates that HexGaze provides a superior balance of speed, efficiency, accuracy, comfort, and learnability when compared to WordPop. Although HexGaze also outperformed Dasher, the difference was not statistically significant, suggesting a trend in

favour of HexGaze but not a conclusive advantage. The technique not only produced stronger performance metrics but also generated more positive experiences, particularly regarding eye fatigue and ease of use. These findings support HexGaze as a promising user-friendly approach for gaze-based text input.

The second experiment allowed us to examine the learning curve of HexGaze across multiple sessions, providing deeper insight into how performance evolves with practice. The results demonstrate a strong and consistent improvement in typing speed across the eight sessions. Users started with an average of 5.59 WPM in Session 1 and reached 9.94 WPM by Session 8, with a steady rising curve in between. This data can be backed up by the statistical analysis that also demonstrates a clear difference between lower sessions and higher ones with the initial sessions being worse. This steady upward curve suggests that users were still improving at the end of the study, meaning the techniques maximum speed potential was not yet reached. In other words, it might be possible to achieve higher speeds with more experience until the curve remains constant.

Throughout all sessions, error rates remained relatively stable, with no meaningful decrease or increase. Statistical analysis confirmed the absence of significant differences across sessions, indicating that errors did not depend on how much users practiced. This stability is likely related to the fact that our participants were not native English speakers, so errors were more tied to uncertainty about spelling than to limitations of the technique itself.

Keystroke efficiency improved notably across sessions. Since keystrokes per character (KSPC) is inversely related to input efficiency, lower values indicate better performance. HexGaze showed a strong decline in KSPC, from 0.770 in the first session to 0.55 in the last, confirming that participants increasingly relied on the word prediction system and required fewer selections to type the same content, typing on average two letters per key selected. Statistical comparisons further supported this behavior, where early sessions had significantly higher KSPC values, while later sessions showed improved efficiency. Similar to the WPM progression, KSPC had not stabilized by the end of the study, suggesting additional efficiency gains could emerge with continued use.

User feedback metrics also improved throughout the study. Comfort, much like the first experiment, started relatively high (average 5.9) and increased steadily to 6.4. Participants reported that as they became familiar with how HexGaze tracked their eye movement, they felt less need to move their head or eyes excessively, which contributed to higher comfort. Early sessions showed more exaggerated head movements from participants trying to avoid incorrect selections, but this behavior diminished as they became more familiar with how eye tracking works.

Perceived typing speed followed the same pattern as WPM performance and it can be seen by comparing both graphs. Ratings increased from 5.7 in Session 1 to 6.4 in the final session, with a steady rise in between and some minor decreases in some sessions, mirroring WPM. Suggesting that HexGaze provides strong user awareness of performance where users can reliably sense when they are typing faster or slower, reinforcing the natural and intuitive feel of the technique.

Eye fatigue however remained stable throughout the study, starting at an average of 4.8 and increasing only slightly to 5.2 by Session 8. The minimal variation and lack of rising discomfort

indicate that HexGaze does not produce much eye strain over prolonged use. Participants likely required fewer eye movements in later sessions due to improved familiarity, which may also explain the slight reduction in perceived fatigue near the end.

Finally, ease of writing improved consistently during the initial sessions. Scores rose from 5.7 in Session 1 to 6.5 by Session 5 and remained stable for the rest of the experiment. This suggests that users require several sessions to fully understand the technique's interaction mechanics, but once they do, writing becomes smooth, intuitive, and effortless. The early increase aligns with improvements in both speed and efficiency, confirming that HexGaze is not only learnable but becomes easier and more pleasant to use rapidly.

Overall, the second experiment demonstrates that HexGaze has a strong and promising learning curve. Users typed faster, with fewer keystrokes, and with greater comfort over time, all while maintaining a stable error rate and very low levels of eye fatigue. Combined with the low precision of the eye tracker required and the technique's low mental and physical load, these findings reinforce that HexGaze is an efficient, easy to learn, and sustainable text entry method suitable for longer typing sessions and capable of achieving high performance with practice.

4.4 Summary

In this chapter, we presented and discussed the two user studies conducted to evaluate HexGaze. The first experiment compared our technique with two gaze-based text entry methods, WordPop and Dasher. This study included 21 participants, although the data from one user had to be removed due to eye tracking inaccuracies when using the Tobii Eye Tracker 4C. The experiment consisted of one session per participant, divided into three parts, each corresponding to one technique. In each part, participants completed a training phase to familiarize themselves with the method, followed by writing eight phrases using that technique.

We then presented the results of the first experiment. HexGaze demonstrated the best overall performance among the three techniques. It achieved the highest typing speed (5.96 WPM), the lowest error rate (0.021 MSD), and better keystroke efficiency than Dasher (WordPop's data had to be removed due to inconsistencies). The competing techniques underperformed due to their respective limitations. Dasher suffered from a steep learning curve and a high likelihood of unintended errors, while WordPop was affected by a poorly positioned space key that increased accidental selections. User feedback aligned with the quantitative results, showing that HexGaze was perceived as more comfortable, less fatiguing, easier to learn, and easiest to type with, making it the preferred technique.

The second experiment evaluated HexGaze over an extended period to characterize its learning curve and understand how performance evolves with repeated use. This study involved 11 participants, with one dataset removed due to a long interruption between sessions. The experiment consisted of eight sessions, with two sessions per day, and in each session participants wrote eight phrases. Data collection was performed using the Tobii Eye Tracker 4C.

We then discussed the results of the second experiment. Typing speed increased from 5.59

WPM at the beginning to 9.94 WPM at the end, showing strong progressive improvement and still increasing, suggesting that users had not yet reached the technique's maximum potential. Error rates remained stable throughout the sessions, likely influenced more by participants' English fluency than by the technique itself. Keystrokes per character (KSPC) improved consistently, decreasing from 0.770 to 0.55, indicating that participants learned to rely more effectively on word prediction and type more efficiently. User feedback metrics also improved: comfort increased across sessions, perceived speed rose in line with actual performance, eye fatigue remained low and stable, and ease of writing increased notably in the first sessions before stabilizing. These findings show that HexGaze is not only learnable and efficient, but also comfortable, with low eye strain while achieving high speeds and low error rates.

Overall, this chapter demonstrated that HexGaze performs strongly both in direct comparison with other techniques and across prolonged usage, offering advantages in speed, efficiency, comfort, and user experience.

Chapter 5

Conclusion

5.1 Summary and Conclusion

In Chapter 1, we introduced the motivation for this work and the main goals of the project. We briefly explained the limitations of traditional eye-typing systems and the need for a technique that is comfortable, accurate, and suitable for low-cost eye trackers while still achieving good performance.

In Chapter 2, we introduced the core concepts related to eye-based interaction, we described the main types of eye trackers, and discussed some of the challenges that affect gaze input, such as tracking imprecision and user fatigue. We also presented the most common selection methods used in gaze interaction, along with their advantages and limitations. Next, we explained the performance metrics used throughout the evaluation, including typing speed (WPM), error rates (MSD), and efficiency measures (KSPC). Finally, we reviewed several existing gaze-based text entry techniques, discussing their designs, strengths, and weaknesses.

In Chapter 3, we presented our proposed technique, which is based on a hexagonal keyboard layout designed for gaze interaction. The system includes basic feedback mechanisms to help users understand their actions and a word prediction component that supports faster text entry. The chapter described the design decisions and how these elements work together to provide a smooth typing experience.

In Chapter 4, we presented our two studies: one comparing HexGaze with two existing techniques and the other evaluating user improvement over time. In the first study, 20 participants completed one session with HexGaze, WordPop, and Dasher, typing eight phrases per technique. HexGaze outperformed WordPop in typing speed, error rate, and overall efficiency. Even though HexGaze also outperformed Dasher in these metrics, there was not a statistically different difference, suggesting a trend in favour of HexGaze but not a conclusive advantage over Dasher. In the second study, 10 participants completed eight sessions with HexGaze. Typing speed and efficiency increased steadily, error rates remained low, and eye fatigue did not increase across sessions.

Based on the results from both studies, we can conclude that HexGaze is an effective and user-friendly gaze-based text entry technique. It reaches a good balance between speed, accuracy, and comfort, thus achieving our goals. HexGaze showed better performance than WordPop and Dasher

for new users, who typed faster, made fewer mistakes, and found the technique more intuitive and comfortable. In the second study, users continued to improve, reaching an average of 9.94 WPM, with half of them achieving average speeds above 10 WPM and one participant achieving max of 17.4 WPM. Error rates stayed low, and eye fatigue remained stable, suggesting that the interface supports extended use while keeping typing speeds high. Our layout, centered on the screen, also works well with low-cost eye trackers, reducing large eye movements and maintain accuracy.

Overall, HexGaze stands out as a practical and well-designed solution for gaze-based typing that addresses common challenges and offers a suitable method for users.

5.2 Future Work

For future work the technique can be improved in some ways we seem could make it faster and with better quality of life upgrades. One idea is to use AI to predict words, taking into account the whole context of the sentence instead of just the last word. This could make predictions more accurate and help users type faster.

Another improvement could be to adjust the selection time for words in the word list hexagon based on word length. Short words could be selected faster with smaller dwell time, while longer words would have the current dwell time to give users a chance to read them properly.

The layout of the hexagon could also be changed to start from the top-right corner instead of the right-center. This would place the word list in the top-left, making it easier for users to see the word they need to type and compare it with the hexagon words. It could also reduce accidental selections that happened in the top-left segment during testing. These changes could make typing smoother, reduce errors, and improve the overall experience.

Finally, user studies can be conducted with people with disabilities. Now that the technique has been tested and shown to be viable and effective for eye typing and thus achieving our goals, the focus can shift toward users with disabilities to evaluate whether the results remain valid for this population and to gather feedback on potential improvements that better address their needs.

Bibliography

[1] M Apte, YYY Agarwadkar, S Azmi, and AB Inamdar. Understanding grids and effectiveness of hexagonal grid in spatial domain. *International Journal of Computer Applications*, 1:25–27, 2012.

[2] Tanya Bafna, Per Bækgaard, and John Paulin Paulin Hansen. Eyetell: Tablet-based calibration-free eye-typing using smooth-pursuit movements. In *ACM Symposium on Eye Tracking Research and Applications*, ETRA '21 Short Papers, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Tanya Bafna-Rührer, Per Bækgaard, and John Paulin Hansen. Smooth-pursuit performance during eye-typing from memory indicates mental fatigue. *Journal of Eye Movement Research*, 15(4):1–16, 2022.

[4] Sam Belina. The future of typing: The hexagonal keyboard layout. `https://producingcuriosity.com/hexagonal-keyboard-intro`, 2024.

[5] Markus Bindemann. Scene and screen center bias early eye movements in scene viewing. *Vision Research*, 50(23):2577–2587, 2010. Vision Research Reviews.

[6] H. Cecotti, Y. K. Meena, and G. Prasad. A multimodal virtual keyboard using eye-tracking and hand gesture detection. *40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3330–3333, 2018.

[7] J. Chung and Jose M. A. Tanchoco. Layout design with hexagonal floor plans and material flow patterns. *International Journal of Production Research*, 48:3407 – 3428, 2010.

[8] Chantal Coles-brennan, Anna Sulley, and Graeme Young. Management of digital eye strain. *Clinical and Experimental Optometry*, 102(1):18–29, 2019. PMID: 29797453.

[9] Michael Cross, Leping Qiu, Mingyuan Zhong, Yuntao Wang, and Yuanchun Shi. One-dimensional eye-gaze typing interface for people with locked-in syndrome. *UIST '22 Adjunct: Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, (43):1 – 3, 10 2022.

[10] Francisco Dias Cardoso. Wordpop: Introducing text using only gaze. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2022.

[11] Piercarlo Dondi, Samuel Sapuppo, and Marco Porta. Leyenes: A gaze-based text entry method using linear smooth pursuit and target speed. *International Journal of Human-Computer Studies*, 184:103204, 2024.

[12] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th annual ACM symposium on user interface software & technology*, pages 457–466, 2015.

[13] Sarah Ezekiel and Google Creative Lab. Look to speak. `https://experiments.withgoogle.com/looktospeak`, 2020.

[14] Wenxin Feng, Jiangnan Zou, Andrew Kurauchi, Carlos H. Morimoto, and Margrit Betke. Hgaze typing: Head-gesture assisted gaze typing. *ETRA '21: ACM Symposium on Eye Tracking Research and Applications*, (11).

[15] Carlos H. Morimoto and Arnon Amir. Context switching for fast key selection in text entry applications. *ETRA '10: Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, pages 271 – 274, 03 2010.

[16] Aleesha Hamid and Per Ola Kristensson. 40 years of eye typing: Challenges, gaps, and emergent strategies. *Proceedings of the ACM on Human-Computer Interaction*, pages 1 – 19, 05 2024.

[17] Thorsten Hansen, Lars Pracejus, and Karl R. Gegenfurtner. Color perception in the intermediate periphery of the visual field. *Journal of Vision*, 9(4):26–26, 04 2009.

[18] Katarzyna Harezlak, Pawel Basek, and Pawel Kasprowski. Side keyboard – the new approach for eye-typing. *Procedia Computer Science*, 207:3348–3357, 2022.

[19] Katarzyna Harezlak, Pawel Basek, and Pawel Kasprowski. Adaptive dwell time for eye typing. *Procedia Computer Science*, 225:4015–4023, 2023. 27th International Conference on Knowledge Based and Intelligent Information and Engineering Sytems (KES 2023).

[20] Henna Heikkilä and Kari-Jouko Räihä. Speed and accuracy of gaze gestures. *Journal of Eye Movement Research*, 3(2):1–14, 2009.

[21] Anke Huckauf and Mario Urbina. Gazing with peye: New concepts in eye typing. *APGV '07: Proceedings of the 4th symposium on Applied perception in graphics and visualization*, page 141, 07 2007.

[22] David J. Ward, Alan F. Blackwell, and David J. C. MacKay. Dasher - a data entry interface using continuous gestures and language models. *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 129 – 137, 11 2000.

[23] Yi Liu, Chi Zhang, Chonho Lee, Bu-Sung Lee, and Alex Qiang Chen. Gazetry: Swipe text typing using gaze. *OzCHI '15: Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*, pages 192 – 196, 12 2015.

[24] I. Scott MacKenzie and R. William Soukoreff. Phrase sets for evaluating text entry techniques. *CHI EA '03: CHI '03 Extended Abstracts on Human Factors in Computing Systems*, pages 754 – 755, 04 2003.

[25] I. Scott MacKenzie and Xuang Zhang. Eye typing using word and letter prediction and a fixation algorithm. *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 55 – 58, 03 2008.

[26] Päivi Majaranta, Ulla-Kaija Ahola, and Oleg Špakov. Fast Gaze Typing with an Adjustable Dwell Time. *CHI '09: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 357 – 360, 04 2009.

[27] Päivi Majaranta, I. Scott MacKenzie, Anne Aula, and Kari-Jouko Räihä. Auditory and visual feedback during eye typing. *CHI EA '03: CHI '03 Extended Abstracts on Human Factors in Computing Systems*, pages 766 – 767, 04 2003.

[28] Mark A. Mento. Different kinds of eye tracking devices. `https://www.bitbrain.com/blog/eye-tracking-devices`, 2020.

[29] Aunnoy K Mutasim, Mohammad Raihanul Bashar, Christof Lutteroth, Anil Ufuk Batmaz, and Wolfgang Stuerzlinger. There is more to dwell than meets the eye: Toward better gaze-based text entry systems with multi-threshold dwell. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–18, 2025.

[30] Omar Namnakani, Yasmeen Abdrabou, Jonathan Grizou, Augusto Esteves, and Mohamed Khamis. Comparing dwell time, pursuits and gaze gestures for gaze interaction on handheld mobile devices. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[31] Antje Nuthmann and George L. Malcolm. Eye guidance during real-world scene search: The role color plays in central and peripheral vision. *Journal of Vision*, 16(2):3–3, 01 2016.

[32] World Health Organization. World health statistics 2024: monitoring health for the sdgs, sustainable development goals. `https://www.who.int/publications/i/item/9789240094703`, 2024.

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

[34] Diogo Pedro, Maria Da Graça Pimentel, Amy Wright, and Khai N. Truong. Filteryedping: Design challenges and user performance of dwell-free eye typing. *ACM Transactions on Accessible Computing (TACCESS)*, 6(1):1 – 37, 03 2015.

[35] Jimin Pi, Paul A. Koljonen, Yong Hu, and Bertram E. Shi. Dynamic Bayesian Adjustment of Dwell Time for Faster Eye Typing. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(10):2315 – 2324, 10 2020.

[36] Marco Porta, Piercarlo Dondi, Alice Pianetta, and Virginio Cantoni. Speye: A calibration-free gaze-driven text entry technique based on smooth pursuit. *IEEE Transactions on Human-Machine Systems*, 52(2):312–323, 2022.

[37] Kari-Jouko Räihä, Päivi Majaranta, and Anne Aula. Effects of feedback on eye typing with a short dwell time. *ETRA '04: Proceedings of the 2004 symposium on Eye tracking research & applications*, pages 139 – 146, 03 2004.

[38] Kari-Jouko Räihä and Saila Ovaska. An exploratory study of eye typing fundamentals: dwell time, text entry rate, errors, and workload. *CHI '12: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3001–3010, 05 2012.

[39] Korok Sengupta, Raphael Menges, Chandan Kumar, and Steffen Staab. Gazethekey: Interactive keys to integrate word predictions for gaze-based text entry. *IUI '17 Companion: Companion Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 121 – 124, 03 2017.

[40] Sima Soltani and Amin Mahnam. A practical efficient human computer interface based on saccadic eye movements for people with disabilities. *Computers in Biology and Medicine*, 70:163–173, 2016.

[41] Magne; Aarås Arne; Kvikstad Tor Martin; Lindberg Lars Göran; Horgen Gunnar Thorud, Hanne-Mari Schiøtz; Helland. Eye-related pain induced by visually demanding computer work. *Optometry and Vision Science*, 89(4):452 – 464, 04 2012.

[42] Mario H Urbina and Anke Huckauf. Dwell time free eye typing approaches. In *Proceedings of the 3rd Conference on Communication by Gaze Interaction (COGAIN 2007)*, pages 65–70, 2007.

[43] Andrew B Watson. High frame rates and human vision: A view through the window of visibility. *SMPTE Motion Imaging Journal*, 122(2):18–32, 2013.

[44] Shu Wei, Desmond Bloemers, and Aitor Rovira. A preliminary study of the eye tracker in the meta quest pro. In *Proceedings of the 2023 ACM International Conference on Interactive Media Experiences*, IMX '23, page 216–221, New York, NY, USA, 2023. Association for Computing Machinery.

[45] Jacob O. Wobbrock. Measures of text entry performance. https://www.sciencedirect.com/topics/computer-science/total-error-rate#:~:text=We%20can%20compute%20an%20MSD,%7C%20%2C%20%7C%20T%20%7C%20)%20., 2007.

[46] Jacob O. Wobbrock, James Rubinstein, Michael W. Sawyer, and Andrew T. Duchowski. Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications*, ETRA '08, page 11–18, New York, NY, USA, 2008. Association for Computing Machinery.

[47] B. R. Wooten and George Wald. Color-vision mechanisms in the peripheral retinas of normal and dichromatic observers. *Journal of General Physiology*, 61(2):125–145, 02 1973.

[48] Zhe Zeng, Elisabeth Sumithra Neuer, Matthias Roetting, and Felix Wilhelm Siebert. A one-point calibration design for hybrid eye typing interface. *International Journal of Human–Computer Interaction*, 39(18):3620–3633, 2023.

[49] Nuowen Zhang, Jing Zhang, Shangsong Jiang, and Weijia Ge. The effects of layout order on interface complexity: An eye-tracking study for dashboard design. *Sensors*, 24(18), 2024.