Gaze Controlled Keyboard Inspired by Steady-State Visually Evoked Potential Methods

BRIAN TAN* and SEPTIA RANI*, Department of Computer Science, Colorado State University, USA

In this project, we aim to create a simple gaze-controlled keyboard that will be accessible to the public. The method is inspired by the Steady-State Visually Evoked Potential (SSVEP), which are signals that are natural responses to visual stimulation at specific frequencies. However, it should be noted that the system that we create is not a Brain-Computer Interface (BCI). Our main goal is to build a simple yet useful gaze-controlled keyboard that can be accessed by anyone, specifically those with speech and mobility impairments. From the user study, we obtained 2.743 seconds as the time needed to scan the letters in the proposed keyboard, which will help avoid the Midas touch problem. We also found that the average time to type is 20.24 seconds per character when the user types using the proposed keyboard.

CCS Concepts: • **Human-centered computing** → **Accessibility technologies**; **Human computer interaction (HCI)**; • **Computing methodologies** → *Cluster analysis*.

Additional Key Words and Phrases: gaze detection, SSVEP, virtual keyboard, clustering

ACM Reference Format:
Brian Tan and Septia Rani. 2024. Gaze Controlled Keyboard Inspired by Steady-State Visually Evoked Potential Methods. *J. ACM* 37, 4, Article 111 (March 2024), 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Brain-Computer Interface (BCI) is a field that is widely known by many Human-Computer Interaction (HCI) researchers. BCI enables a person to control an external device using brain signals [16]. BCI can be a very helpful application to human systems. Specifically, it can be very helpful to those with physical disabilities [8, 14, 15]. In this paper, we would like to specifically address how it may be helpful to those with speech and mobility impairments.

BCI has been around for a long time, there are already many advances to BCI for communication [9, 20]. One of the currently most used paradigms for communication is called Steady-State Visually Evoked Potential (SSVEP) [18], which are signals that are natural responses to visual stimulation at specific frequencies. The current state-of-the-art non-invasive virtual keyboard system using SSVEP produces around 12 words per minute [2]. Currently, many researchers are still working on improving this system.

Although BCI has been very helpful for communication systems, there are two main problems that researchers face when developing this system. First, the hardware needed for BCI systems is expensive [24]. Same as most hardware

*Both authors contributed equally to this research.

Authors' address: Brian Tan, brian.tan@colostate.edu; Septia Rani, septia.rani@colostate.edu, Department of Computer Science, Colorado State University, 456 University Ave 444, Fort Collins, Colorado, USA, 80521.

Manuscript submitted to ACM

related research, there is always a cost-accuracy tradeoff that we need to worry about. Second, data from BCI systems are very complex [22]. The raw brain signal decoded from the sensor is too complex for simple learning methods to recognize any pattern. This is mainly due to the noise accumulated from non-invasive sensors [4].

To successfully build a fully functional BCI system, we have to understand some complicated methods to be used on our data. These methods can include Fourier Analysis, Principal Component Analysis, and many more [17]. It can also be noted that many experiments conducted on BCI require a high cognitive load that can usually be achieved by participants in a quiet lab environment. However, that load might not be attainable in a real-world environment where there are many distractions [17].

An alternative method to build a virtual keyboard is by using an eye tracker. We can use an eye tracker to detect what letter we are trying to spell. However, this method also comes with its problems. A computer vision-based eye tracking system is very sensitive to many factors such as degree of eye openness, variability in eye size, head pose, etc [1]. Therefore, to accurately map gaze directions from one's eye to a computer screen, it must be a condition that the eye position cannot move at all relative to the camera.

A fix to this problem is to use eye-tracking glasses that have a camera attached to the glasses. This would eliminate the problem of external factors. Once calibrated, the systems would easily map without a problem. However, eye-tracking glasses are expensive, and not everyone can get access to them.

We propose to build a system that uses techniques inspired by both SSVEP and eye tracking that can be accessible to anyone. We will be using a *partitioning* system commonly used in SSVEP systems [13]. It should be noted that the system that we create is not a BCI. This system equally partitions the Latin alphabet into four different groups. Once the user selects a group where their desired character is, it will again partition all the characters in the specific group into four new groups. It will keep on repeating until the system breaks it down to a single character per group, and the user finally picks their desired character.

We aim to use a simple approach to eye tracking as a selection method for this system. Since we are only aiming to differentiate between four different commands, we can manage to *relax* our accuracy constraint. We can simply localize our eye using simple computer vision techniques and then partition it into four different quadrants. The four different quadrants include top-left, top-right, bottom-left, and bottom-right to represent Command 1, Command 2, Command 3, and Command 4 respectively.

2 RELATED WORKS

Our work is exclusively based on common techniques and paradigms known in the field of BCI and eye tracking. The problems we face are also common problems in previous literature.

2.1 Steady-State Visually Evoked Potential

Steady-State Visually Evoked Potential (SSVEP) is a widely known paradigm in the field of BCI [18]. In the center-back part of our brain, we have what is called the Occipital lobe, which processes the visual signals from our eyes [23].

SSVEP uses flickering lights, with its frequency ranging anywhere above 4 Hz [5], to send a signal to the occipital lobe as an event. The commonly used sensor used for SSVEP is called an Electroencephalography (EEG) sensor. Figure 1 shows a standard 20-channel node placement for EEGs. For SSVEP systems, we only care about the O1, O2, and OZ nodes. SSVEP is usually paired with a Graphical User Interface (GUI) containing multiple boxes flickering at different frequencies to represent each command.

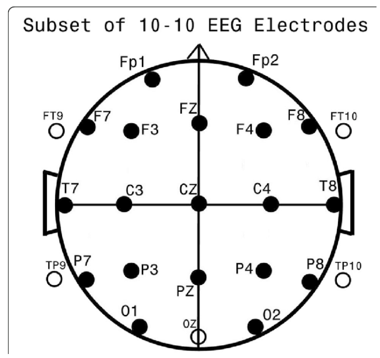

Fig. 1. Standard 20 channel EEG node placement [6]

We have established that the problems with SSVEP systems are cost and complexity. Data analysis of the three-dimensional nodes can be very complex if not done right. For accessibility's sake, we have decided that for this project, we aim to use only the GUI multi-command partitioning technique used for SSVEP. We can now eliminate both problems for our system.

2.2 Eye/Gaze Tracking

Eye tracking has been a widely known paradigm for a while now. It has been accessible to the public, but still at an expensive price [19]. To build an eye tracking system locally using a camera, it is a condition that our head must stay exactly still the whole time [1], which is not possible nor realistic. Even if we can achieve stillness, processing gaze mapping is a very complex system that requires us to master the math behind vector projections and rotations.

A workaround for this problem is to use eye-tracking glasses so that our eyes move along with the camera, which implies no relative movements. However, eye-tracking glasses are expensive and are still inaccurate.

We realized that the need for such extreme conditions is to be able to get an accurate mapping up to every single pixel. However, since we just want to differentiate between four different commands, we do not need such accuracy. We can *relax* these conditions since we do not need extreme accuracy. We can simply localize our eye using simple computer vision techniques and then figure out which quadrant our pupil is in. Figure 2 gives a clear illustration of how to differentiate between the quadrants.

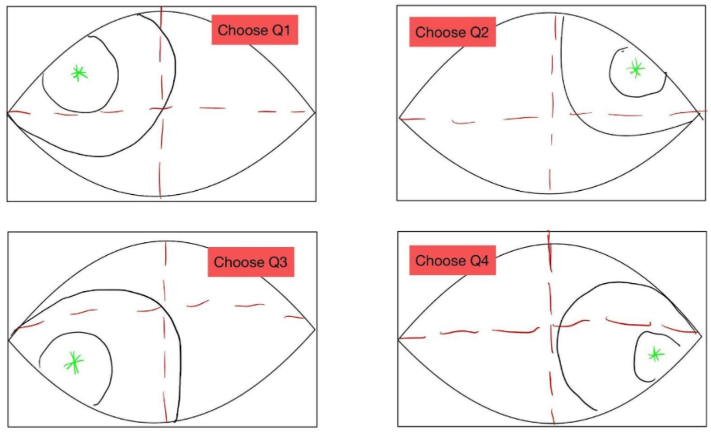

Fig. 2. Illustration of the “eye-tracking” to choose which quarter to go

2.3 Partitioning Tree

Now that we have defined a notion of commands (GUI) and selection methods (Gaze), we can start defining how we can use what we have to build a keyboard. This system can be used for any set to be partitioned. We will always get a degree-four-bounded tree that can be used to cover all the cases. Since our goal is to build a keyboard, we will have the following tree shown in Figure 3.

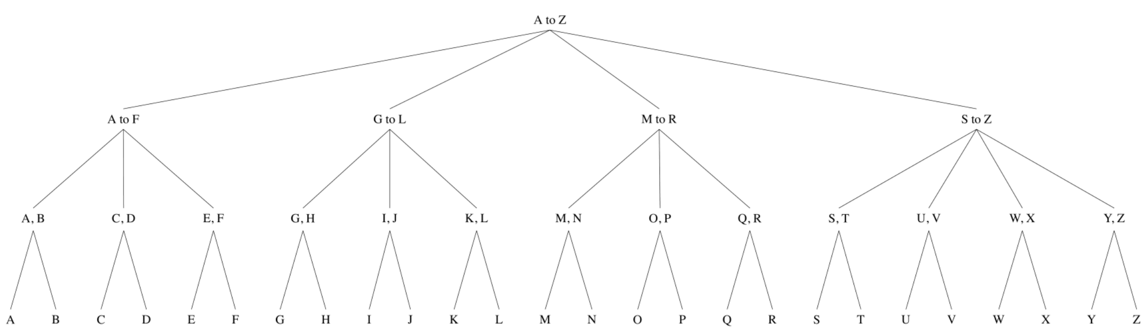

Fig. 3. Partitioning tree of the Latin alphabet

Since our keyboard only consists of the Latin alphabet, our tree will be finite with a depth of three. For future development, a finite state space will always result in a degree-four-bounded tree with depth $\lceil \log_4(n) \rceil$ where n is the size of the set to be partitioned on. This gives the benefit of scanning only $\lceil \log_4(n) \rceil$ items instead of a linear scan.

2.4 Midas Touch

Midas Touch is a well-known problem in HCI [7]. To avoid Midas Touch as much as we can, we decided to conduct an experiment to find the time it takes to scan through our proposed keyboard. The details of this experiment will be explained in our methodology. Let T_{total} be the time needed to avoid Midas Touch. T_{total} can be expressed as

$$\inf \{t > 0 \mid t \text{ avoids Midas Touch}\} \tag{1}$$

Since the time to scan a set of items is monotonic relative to its length, then the time it takes to scan through the first layer of our GUI will take the most time out of all the layers. Let T_{scan} be the time it takes to scan through the alphabet in layer 1. We will be determining what T_{scan} is in the experiment stated earlier.

Recent research has shown that when a participant is asked to select something through a gaze interface, a *dip* is formed in the EEG to confirm that they intend to select that specific item [21]. The paper showed that a dip is usually formed around 200 ms after staring at the desired object. We can associate this dip time as confirmation time. Let $T_{\text{confirm}} = 200$ms.

Using the information above, we can have an upper bound on T_{total}

$$T_{\text{total}} \leq T_{\text{scan}} + T_{\text{confirm}} = T_{\text{scan}} + 200ms \tag{2}$$

We hypothesize that upper bounding T_{total} will help avoid Midas Touch.

3 METHODOLOGY

3.1 Participants and Experiment Design

The study includes two parts. First, we conduct a user study to measure the T_{scan}. After we get the T_{scan} from the experiment, we will use that to conduct another user study to measure the performance of the Gaze Controlled Keyboard prototype that we propose.

For the first user study, we recruit twenty participants (10 females and 10 males). Their ages ranged between 18 and 42. In general, there are no specific requirements for the participants. The interface of the application that we use to conduct the user study can be seen in Figure 4.

The application has four different quadrants (Q1, Q2, Q3, and Q4), each containing 6-8 different letters from the alphabet. For example, the letter "A", "B", "C", "D", "E", and "F" can be found in Q1, and the letter "M", "N", "O", "P", "Q", and "R" can be found in Q3. In this experiment, for each participant, we will generate 12 different letters (12 trials), three from each quadrant. For each letter, we will record the time needed for the participant to find which quadrant contains the letter.

To start the experiment, the participant needs to press the "Start" button at the beginning of every trial. The timer will then start. After this, the participant will be instructed to find the given letter on the current screen, e.g., find "E." The participant needs to search for which quadrant contains the letter. Here, we can find "E" in Q1. The participant needs to press button "1" on the keyboard.

The trial will be repeated 12 times. The first four trials are designed to search for one random letter from each quadrant, and the rest will be in random order. This scenario will let the participant learn to search for letters from

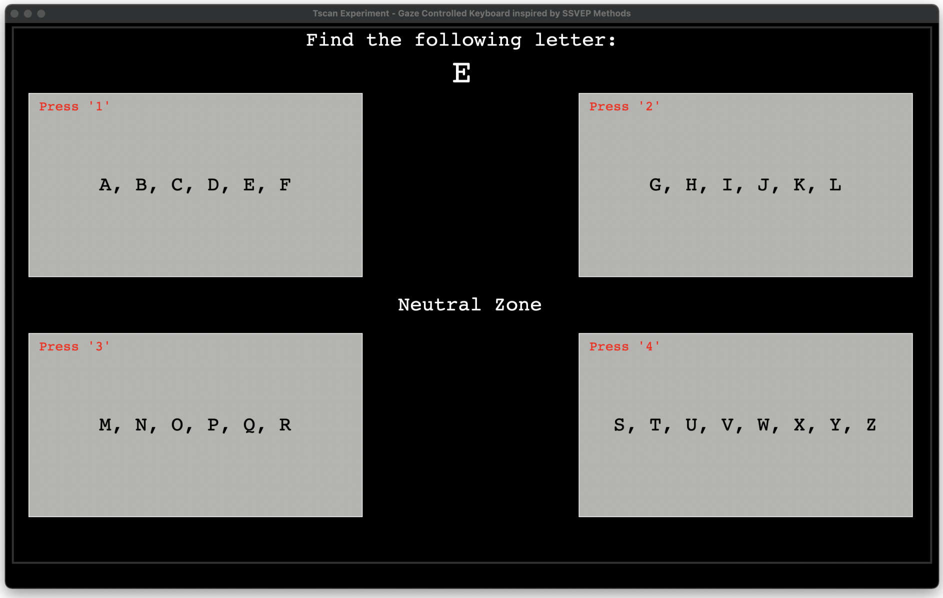

Fig. 4. Screenshot of the application to measure the T_{scan}

each quadrant. Finally, after we finish with 12 trials, we will extract the record from the experiment. We can know what key is pressed by the participant. We can compare it with the number of the ground truth quadrant, whether it is correct or not. For example, if the letter to find is "E", and the participant presses "1" on the keyboard, that is correct. But if given "E" to search, and the participant presses "2" on the keyboard, that is incorrect. We also collect the time needed to scan the letter given in each trial. The time needed to scan is calculated from the second when the instruction to find a letter appears on the screen until the participant confirms by pressing the button on the keyboard. From all of the data collected in the first user study, we are trying to infer the T_{scan}. We will use T_{scan} to estimate the time needed to avoid the Midas touch problem for the Gaze Controlled Keyboard that we propose in this paper.

The second user study is intended to measure the performance of the Gaze Controlled Keyboard prototype that we propose. We recruited five participants (1 female and 4 males). Each participant is asked to type a specific word, in this case, we use the word "HELLO" for all the participants. Before the participant types using the Gaze Controlled Keyboard, there is a calibration step that intends to find a boundary suitable for each person. The details of this step are explained further in subsection 3.2.

From this user study, we collect the time needed to type the "HELLO" word for each participant. We also record the words typed by the participant to obtain accuracy. We will use the time and accuracy to evaluate the performance of our prototype.

For both user studies that we conducted, the program was run on a MacBook Pro (14-inch) laptop equipped with an Apple M1 Pro chip and 16 GB memory. To run the applications, we need Python 3.11 to be installed on the laptop.

3.2 Gaze Tracking and Calibration

In this subsection, we will describe the methods used for the gaze selection method in detail.

3.2.1 Computer Vision. We used a webcam-based eye-tracking library created by MIT [11]. The library uses simple Computer Vision techniques to locate the center of the pupil and return a vertical and horizontal ratio with respect to the eye frame. We can treat these ratios as x, y coordinates with range $[0, 1]$.

3.2.2 Data Collection. In an ideal world, the boundaries of the x, y coordinates can be easily differentiated by $(0.5, 0.5)$. However, every person's eye frame and gaze are different; thus, we need to find a boundary suitable for each person. Before users can start using the software, they are asked to go through a calibration phase. We ask each user to stare at the center of each quadrant, and we will record the x, y coordinates for 5 seconds. We will repeat the process for the four quadrants. Figure 5 shows a screenshot of how the data collection would look like.

Fig. 5. Screenshot of the data collection

3.2.3 K-Means Clustering. After we finish our data collection, we should get a 2−Dimensional vector mapped (coordinates) to a one-dimensional vector (labeled quadrant). Since we want to differentiate between four clusters of data, we can simply use k−means clustering for efficient computational cost [12]. Once we have trained our model from our calibration dataset, every iteration of the main loop will use this model to determine which quadrant we are currently staring at. One problem that we will solve in the next part is that k−means clustering is an unsupervised machine learning algorithm. This would mean that the label produced by the algorithm might not follow the same sequence as the actual labels.

3.2.4 Relabeling. As we have established earlier, k−means clustering is an unsupervised machine learning algorithm that makes the resulting labeling not in the right order. We can fix this problem by connecting it to a famous combinatorial optimization problem called the Linear Sum Assignment Problem (LSAP) [10]. The LSAP can be formally defined as follows:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A\times B} C_{ij} \cdot x_{ij} \\ \text{s.t.} \quad & \sum_{j\in B} x_{ij} = 1 \\ & \sum_{i\in A} x_{ij} = 1 \\ & \forall i,j \in A\times B,\ x_{ij} = \{0,1\},\ C_{ij} \in \mathbb{R}_{\geq 0} \end{aligned} \tag{3}$$

In simple terms, we have two sets A, B, and a cost function $C : A \times B \longrightarrow \mathbb{R}_{\geq 0}$. Our objective is to find a *bijection* from A to B with minimal cost. Since we want to reorder the label from the $k-$means model to the right order, this is equivalent to finding a bijection from the $k-$means cluster centroids to the recorded data centroids with our cost function to be Euclidian distance [3]. This will pair up each cluster centroids with its nearest recorded data centroids, thus relabeling it correctly.

3.2.5 Calibration Results. After we are finished with data collection, clustering, and relabeling, we will have a calibrated model suited for each user to determine which quadrant a user is currently staring at. The example of the calibration result can be seen in Figure 6

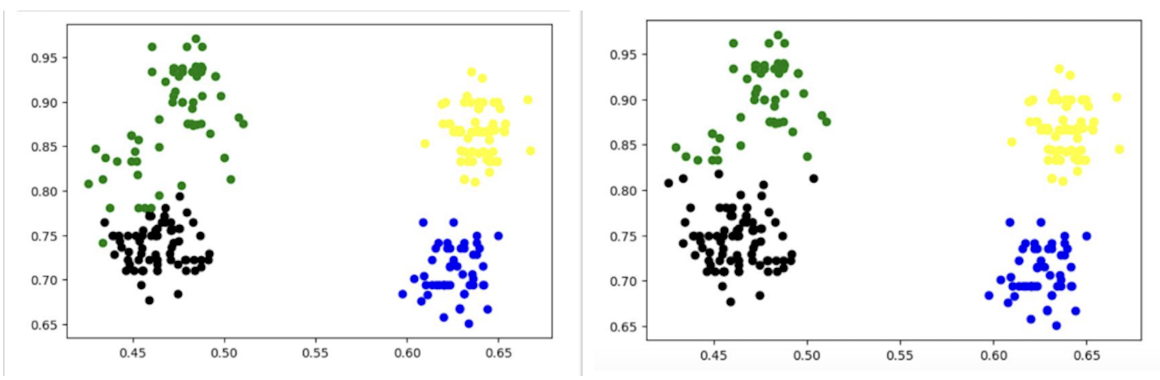

Fig. 6. Recorded data (left) and $k-$means clustering + relabeling (right)

4 RESULTS

4.1 T_{scan} User Study

The result of the T_{scan} user study can be seen in Figure 7. Each graph represents the line charts of the time to scan needed for each participant for each quadrant (Q1, Q2, Q3, and Q4). As mentioned earlier, each participant did 12 trials, 3 trials (Trial 1, Trial 2, and Trial 3) for each quadrant. From the trend lines that are denoted by the thick black lines, we can see that, in general, the time needed to scan the letters in the proposed Gaze Controlled Keyboard is getting shorter. It shows that there is learning progress demonstrated by the participants.

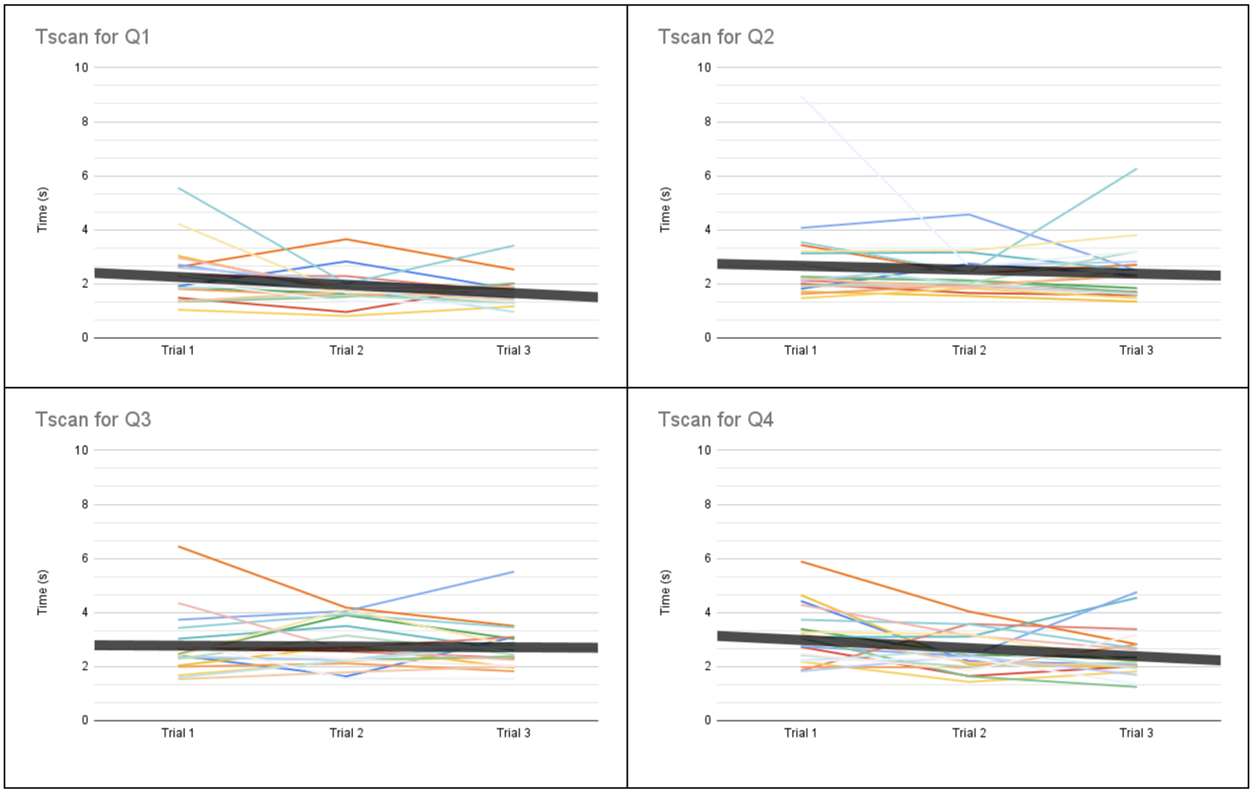

Fig. 7. Result of the T_{scan} user study

In addition, we also calculate the average of T_{scan} for each quadrant. The result can be seen in Figure 8. For each quadrant, we also calculate and plot the standard deviation, whose values range from 0.804 to 1.198. From the figure, we obtain 2.743 s as the highest average of T_{scan}. We will use this value in the proposed Gaze Controlled Keyboard to help avoid Midas Touch problem.

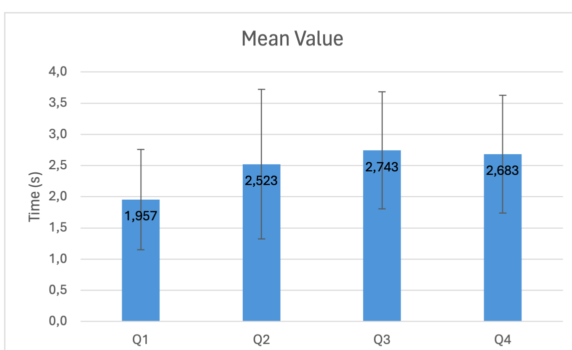

Fig. 8. The average of the T_{scan} for each quadrant

Table 1. The summary of the accuracy and time performance for the Gaze Controlled Keyboard

Trial	Word typed	Accuracy (%)	Time to type (s)
1	HELLA	80	129
2	HELLO	100	83
3	HELLO	100	136
4	HELLO	100	93
5	HELLO	100	65
	Average	96	101.2

4.2 Gaze Controlled Keyboard Performance

We conducted a user study to measure the performance of the Gaze Controlled Keyboard prototype. All participants were asked to type the "HELLO" word. Then, we measured the accuracy and time needed for each participant to type that word. The formula to calculate the accuracy is as follows.

$$Accuracy = \frac{\text{The number of correct characters}}{\text{The number of characters in the word}} \tag{4}$$

The summary of the result can be seen in Table 1. From the experiments, the average accuracy is 96%, and the average time to type is 101.2 s, which gives 20.24 s per character.

5 DISCUSSIONS

The proposed system simplifies the user interaction process by partitioning the Latin alphabet into four groups. It allows users to select characters through a series of simpler choices. This study also demonstrates a research effort to address the Midas Touch problem, which is a common challenge in gaze-based interfaces where unintentional selection can occur.

From Table 1, we saw that the average time to spell out a five-letter word is around 100 seconds. Since this system is built for accessibility and not general use, so 100 seconds is pretty reasonable. Most of the time taken is because of accidental look-aways, which happen very often, which resets the T_{scan} counter to 0 again.

Another flaw is that the screen size is not big enough to create the significant effect of eye shift that we had hoped to see. This causes the data points to be close to one another which makes the predictive model have a thin boundary. In a real SSVEP system, this would not be an issue as the frequency of the object stared at blurs out the peripheral version. Additionally, the calibration process, though crucial for the system's accuracy, presents a barrier to immediate use, suggesting a need for a more streamlined setup.

Despite the limitations, our research demonstrates the feasibility of a gaze-controlled keyboard that leverages techniques inspired by both SSVEP and eye-tracking technologies. It offers a novel approach with affordable cost to assist individuals with speech and mobility impairments.

6 CONCLUSION AND FUTURE WORK

In this paper, we demonstrated that we can build a simple yet useful gaze-controlled keyboard. The method is inspired by the Steady-State Visually Evoked Potential. We used a simple eye-tracking technique rather than a BCI system. From the user study, we obtained 2.743 seconds as the time needed to scan the letters in the proposed keyboard, which will help avoid the Midas touch problem. We also found that the average time to type is 20.24 seconds per character when the user types using the proposed keyboard.

For future work, we need to investigate more advanced algorithms for the eye-tracking method used in the proposed Gaze Tracking Keyboard to enhance the system's responsiveness and accuracy. By addressing the limitations identified, we can continue to refine and enhance this technology and move closer to our goal of creating inclusive and accessible communication technology.

7 ACKNOWLEDGMENTS

We would like to thank Dr. Francisco Ortega for the valuable feedback on this project.

REFERENCES

[1] Amer Al-Rahayfeh and Miad Faezipour. 2013. Eye tracking and head movement detection: A state-of-art survey. *IEEE journal of translational engineering in health and medicine* 1 (2013), 2100212–2100212. https://doi.org/10.1109/jtehm.2013.2289879

[2] Xiaogang Chen, Yijun Wang, Masaki Nakanishi, Xiaorong Gao, Tzyy-Ping Jung, and Shangkai Gao. 2015. High-speed spelling with a noninvasive brain–computer interface. *Proceedings of the national academy of sciences* 112, 44 (2015), E6058–E6067. https://doi.org/10.1073/pnas.1508080112

[3] D. Cohen. 2004. *Precalculus: A Problems-Oriented Approach.* Cengage Learning. https://books.google.com/books?id=_6ukev29gmgC

[4] Alexandre Défossez, Charlotte Caucheteux, Jérémy Rapin, Ori Kabeli, and Jean-Rémi King. 2023. Decoding speech perception from non-invasive brain recordings. *Nature Machine Intelligence* 5, 10 (2023), 1097–1107. https://doi.org/10.1038/s42256-023-00714-5

[5] Pablo F Diez, Vicente A Mut, Enrique M Avila Perona, and Eric Laciar Leber. 2011. Asynchronous BCI control using high-frequency SSVEP. *Journal of neuroengineering and rehabilitation* 8 (2011), 1–9. https://doi.org/10.1186/1743-0003-8-39

[6] Frank H Duffy, Aditi Shankardass, Gloria B McAnulty, and Heidelise Als. 2017. A unique pattern of cortical connectivity characterizes patients with attention deficit disorders: a large electroencephalographic coherence study. *BMC medicine* 15 (2017), 1–19. https://doi.org/10.1186/s12916-017-0805-9

[7] Robert JK Jacob. 1995. Eye tracking in advanced interface design. *Virtual environments and advanced interface design* 258, 288 (1995), 2. https://www.researchgate.net/publication/2924246_Eye_Tracking_in_Advanced_Interface_Design

[8] Nancy S Jecker and Andrew Ko. 2022. The unique and practical advantages of applying a capability approach to brain computer interface. *Philosophy & Technology* 35, 4 (2022), 101. https://doi.org/10.1007/s13347-022-00597-1

[9] Aleksandra Kawala-Sterniuk, Natalia Browarska, Amir Al-Bakri, Mariusz Pelc, Jaroslaw Zygarlicki, Michaela Sidikova, Radek Martinek, and Edward Jacek Gorzelanczyk. 2021. Summary of over fifty years with brain-computer interfaces—a review. *Brain Sciences* 11, 1 (2021), 43. https://doi.org/10.3390/brainsci11010043

[10] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97. https://doi.org/10.1002/nav.3800020109

[11] Antoine Lame. 2019. Eye Tracking library easily implementable to your projects. https://github.com/antoinelame/GazeTracking?tab=readme-ov-file

[12] James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297. https://api.semanticscholar.org/CorpusID:6278891

[13] Luca Maggi, Sergio Parini, Luca Piccini, Guido Panfili, and Giuseppe Andreoni. 2006. A four command BCI system based on the SSVEP protocol. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society.* IEEE, 1264–1267. https://doi.org/10.1109/iembs.2006.260353

[14] Joseph N Mak and Jonathan R Wolpaw. 2009. Clinical applications of brain-computer interfaces: current state and future prospects. *IEEE reviews in biomedical engineering* 2 (2009), 187–199. https://doi.org/10.1109/RBME.2009.2035356

[15] José del R Millán, Rüdiger Rupp, Gernot Mueller-Putz, Roderick Murray-Smith, Claudio Giugliemma, Michael Tangermann, Carmen Vidaurre, Febo Cincotti, Andrea Kubler, Robert Leeb, et al. 2010. Combining brain–computer interfaces and assistive technologies: state-of-the-art and challenges. *Frontiers in neuroscience* 4 (2010), 1613. https://doi.org/10.3389/fnins.2010.00161

[16] Muhammad F Mridha, Sujoy Chandra Das, Muhammad Mohsin Kabir, Aklima Akter Lima, Md Rashedul Islam, and Yutaka Watanobe. 2021. Brain-computer interface: Advancement and challenges. *Sensors* 21, 17 (2021), 5746. https://doi.org/10.3390/s21175746

[17] Luis Fernando Nicolas-Alonso and Jaime Gomez-Gil. 2012. Brain computer interfaces, a review. *sensors* 12, 2 (2012), 1211–1279. https://doi.org/10.3390/s120201211

[18] Anthony M Norcia, L Gregory Appelbaum, Justin M Ales, Benoit R Cottereau, and Bruno Rossion. 2015. The steady-state visual evoked potential in vision research: A review. *Journal of vision* 15, 6 (2015), 4–4. https://ncbi.nlm.nih.gov/pmc/articles/PMC4581566/

[19] Jakub Štěpán Novák, Jan Masner, Petr Benda, Pavel Šimek, and Vojtěch Merunka. 2023. Eye tracking, usability, and user experience: A systematic review. *International Journal of Human–Computer Interaction* (2023), 1–17. https://doi.org/10.1080/10447318.2023.2221600

[20] Jiahui Pan, XueNing Chen, Nianming Ban, JiaShao He, Jiayi Chen, and Haiyun Huang. 2022. Advances in P300 brain–computer interface spellers: toward paradigm design and performance evaluation. *Frontiers in Human Neuroscience* 16 (2022), 1077717. https://doi.org/10.3389/fnhum.2022.1077717

[21] GS Rajshekar Reddy, Michael J Proulx, Leanne Hirshfield, and Anthony J Ries. 2024. Towards an Eye-Brain-Computer Interface: Combining Gaze with the Stimulus-Preceding Negativity for Target Selections in XR. *bioRxiv* (2024), 2024–03. https://doi.org/10.1101/2024.03.13.584609

[22] Simanto Saha, Khondaker A Mamun, Khawza Ahmed, Raqibul Mostafa, Ganesh R Naik, Sam Darvishi, Ahsan H Khandoker, and Mathias Baumert. 2021. Progress in brain computer interface: Challenges and opportunities. *Frontiers in systems neuroscience* 15 (2021), 578875. https://doi.org/10.3389/fnsys.2021.578875

[23] Suzan Uysal. 2023. The Occipital Lobes and Visual Processing. In *Functional Neuroanatomy and Clinical Neuroscience: Foundations for Understanding Disorders of Cognition and Behavior*. Oxford University Press. https://doi.org/10.1093/oso/9780190943608.003.0014

[24] Athanasios Vourvopoulos and Sergi Bermudez i Badia. 2016. Usability and cost-effectiveness in brain-computer interaction: is it user throughput or technology related?. In *Proceedings of the 7th Augmented Human International Conference 2016*. 1–8. https://doi.org/10.1145/2875194.2875244

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009