QC Speller: User Interface Design of a Hands-Free Touch-Free Speller with Brain Electroencephalogram Sensory Rhythm

TUTAN NAMA and DEBASIS SAMANTA, Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

Nowadays, brain-computer interfaces (BCIs) are being extensively explored by researchers to recover the abilities of motor-impaired individuals and improve their communication. Text entry is one of the most important regular tasks in communication, and BCI has high application potential to develop a speller. Although BCI has been a growing research topic for the last decade, more progress is yet to be made for BCI-based spellers. Two challenges are there in BCI speller development: designing an effective graphical user interface with an optimal arrangement of symbols enabling a minimum number of control commands and reducing users' efforts in error correction following an efficient error correction policy. With this scope in mind, this work proposes a novel BCI speller paradigm with an efficient symbol arrangement to improve the text entry rate. Additionally, it provides a user-friendly error correction approach through six text-cursor navigation keys to enhance the accuracy of text entry. The proposed speller includes 36 target symbols and is operated with two control commands obtained from electroencephalogram motor-imagery signals. The experimental results revealed that the proposed speller outperforms the existing motor imagery-based BCI spellers when tested on 10 motor-impaired users. This speller achieved a mean performance of 5.20 characters per minute without any typing error and 6.04 characters per minute with a 2.9% mean error. The mean correction efficiency was 0.69; that is, users corrected 69% of incorrect inputs with a single correction key pressed.

CCS Concepts: • **Human-centered computing** → **Graphical user interfaces**; **Text input**;

Additional Key Words and Phrases: Hands-free touch-free interaction, User interface design, Brain-Computer Interface (BCI), BCI speller, Text entry system, Assistive text entry, People with motor impairment

ACM Reference format:
Tutan Nama and Debasis Samanta. 2024. QC Speller: User Interface Design of a Hands-Free Touch-Free Speller with Brain Electroencephalogram Sensory Rhythm. *ACM Trans. Access. Comput.* 17, 4, Article 20 (December 2024), 36 pages.
https://doi.org/10.1145/3705733

Tutan Nama has designed the idea, implemented prototype, carried out the experiments, analyzed the results, and drafted the article. Debasis Samanta verified the ideas, reviewed the results and the article, and finally proof read the article.

Authors' Contact Information: Tutan Nama (corresponding author), Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India; e-mail: tutancsephd@iitkgp.ac.in; Debasis Samanta, Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India; e-mail: dsamanta@iitkgp.ac.in.

ACM 1936-7236/2024/12-ART20
https://doi.org/10.1145/3705733

1 Introduction

Motor impairments restrict individuals' communication opportunities and create social distances among people. To bridge this communication gap, several **assistive technologies (ATs)** [10, 21], like mouth sticks, head wands, single-hand keyboards, and others, have been developed. However, most ATs need different forms of physical input to operate [3], which limits their usage for severe motor-impaired people. In contrast, the **brain-computer interfaces (BCIs)** utilize the human brain signals as control inputs to establish communication [5, 10] between humans and computers; thus, BCIs can be used as an alternative text entry support for a person with motor impairment. Generally, BCI systems capture a user's brain waves through different neuroimaging techniques and transform these brain waves into control commands for different applications.

There are three different brain imaging methods, invasive (e.g., intracortical electrode array), semi-invasive (e.g., **electrocorticography (ECoG)**), and non-invasive (e.g., **electroencephalography (EEG)**). In invasive methods, brain signal acquisition is acquired by surgery and planting electrode array into the gray matter of the brain. In semi-invasive, electrodes are placed inside the skull but over the gray matter for brain signals acquisition, hence it has long-term stability than invasive one. Although the invasive and semi-invasive technique provides good quality data with a high spatial resolution, it is a hazardous and costly process, and is very difficult to apply in real-life applications for daily usage [43]. On the other hand, EEG is one of the most popular and safe brain imaging techniques for BCI-based applications, satisfying both convenience criteria and effectiveness criteria [14, 31, 33]. Of late, several EEG signal-based **human-computer interaction (HCI)** applications, such as control modalities for games [20, 32, 39], entertainment based on emotion recognition [22], concentration level recognition [1], and others [2, 19], have been developed. One of the prospective utilization of EEG-based BCI is to develop a cost-effective hands-free touch-free text entry mechanism [7, 8, 11], termed as BCI speller.

To design control modalities for a BCI speller, three types of EEG brain activity patterns [5, 33] have been predominantly used in the literature. These brain signals are event-related synchronization/desynchronization of **sensorimotor rhythms (SMRs)**, **steady-state visual evoked potentials (SSVEPs)**, and P300 components of **event-related potentials (ERPs)** [34]. Depending on the intrinsic nature of these brain signals, the operation modality of a BCI speller can be synchronous or asynchronous. The P300 and SSVEP signals are synchronous and bounded for a pre-defined time frame with external stimuli. Accordingly, in a synchronous speller, the users must depend on external stimuli to excite a command for speller control. In contrast, the **motor imagery (MI)** SMRs are independent of any external stimuli, and the spellers controlled by MI signals are asynchronous [34]; hence, the users may have complete control over BCI spellers. For this reason, the MI-based BCI speller paradigms are recently gaining popularity [34].

Motor imagination is a complex cognitive operation, and it is a challenging task to reliably classify a large set of MI classes. Hence, challenge in MI-based BCI spellers is to map a small number of control commands to a large set of characters or target symbols [34]. In this regard, recently several attempts have been made to develop an effective symbol arrangement and corresponding mapping technique. In [36], 28 target symbols (English letters and two special functions) were arranged in alphabetical order, and 3 control commands (left-hand, right-hand, and feet MI) were used for symbol selection. In [7], a set of 30 target symbols arranged alphabetically and divided into six groups. The groups were placed on six hexagons with the intent to reduce the number of control commands and increase the number of target symbols. In this case, the symbols were mapped with only two control commands (right-hand and right-foot MI). Although these approaches have achieved a considerable text entry rate, the accuracy in text entry is low. One of the main reasons for the low accuracy is inefficient error correction policy and less number of correction operators

for the purpose. In most of the existing BCI spellers, error correction was done using a sequence of backspaces that increase users' effort, leading to lower accuracy. As the text entry rate and text **entry accuracy (EA)** both are essential performance parameters for an efficient speller, there is a need for an optimal error correction policy to increase the accuracy of text entry. Moreover, in the existing spellers, the letters are arranged alphabetically instead of their frequency of use, which increases overall text **input time (IT)** and reduces speller performance.

This study proposed an efficient and novel BCI speller paradigm considering only two control commands, a more extensive set of target symbols with an optimal symbol arrangement scheme, and an efficient error correction policy. In the proposed approach, the speller **graphical user interface (GUI)** consisted of 36 target symbols (26 letters, 2 punctuation marks, space, backspace, and 6 special functions). Symbols were placed over four quarter circles of a circular zone, with nine symbols in each quarter circle. For achieving an optimal symbol arrangement, the letters were arranged according to their frequency of use and grouped effectively using the binary Huffman coding algorithm [16]. In the proposed speller interface, a backspace symbol ("←") and a back operation key ("BACK") have been included for correcting any misspelled letter(s) and undoing an action in second and third stages, respectively. Further, a more sophisticated error correction scheme was introduced for word/phrase level correction by adding six special symbols ("<," ">," "≪," "≫," "⋘," and "⋙"). These special symbols enabled the user to freely navigate the text-cursor on the text-field to reach the incorrect letter faster. To the best of our knowledge, the proposed BCI speller paradigm is the first paradigm that utilizes the text-cursor navigation keys to increase **correction efficiency (CE)** by reducing users' effort in error correction. Later, the proposed speller interface was integrated with a highly accurate sparse representation-based EEG-MI signal classification model [37], and text entry experiments were performed to validate the proposed speller. This work also provides a systematic structure of design and analysis of user interface for any BCI application.

The rest of the article is organized as follows: Section 2 surveys the existing approaches to building MI-based BCI spellers; Section 3 gives an overview of the proposed system. The design process of the proposed speller interface, evaluations of the prototypes, and its working principle are discussed in Section 3.1. Section 3.2 describes the construction of the BCI subsystem, including data acquisition, pre-processing, feature extraction, and classification process. Section 3.3 describes the multi-threading integration mechanism and includes relevant pseudocodes. Section 4 discusses the experiments with users. Section 5 offers the outcomes of the experiment. The significance of the results is discussed in Section 6. Lastly, Section 7 concludes the work.

2 Related Work

Of late, MI-based BCI spellers are being explored by researchers due to their asynchronous nature. In developing any asynchronous BCI speller, researchers focus on two significant parts: one is the back-end part for brain signal processing and other one is the front-end part, a graphical user interface for text formulation. In 2018, Rezeika et al. [34] published a review paper on BCI spellers. They strongly emphasized that the front-end is the first parameter that an end-user would judge on the performance of a BCI speller. This section includes some approaches that contributed to the front-end part of designing MI-based BCI speller.

The first MI-based BCI speller reported in the literature was proposed by Obermaier et al. [28]. The virtual keyboard of this speller was adopted from the two-class slow cortical potential-based speller proposed by Birbaumer et al. [6]. In this speller, 26 letters were arranged alphabetically along with six other symbols and placed in two groups. Two MI classes were used to select two groups distinctly. The selection of a target in this speller requires six mental tasks, and correcting an error using the "DELETE" or "BACK" operation needs seven mental tasks. This speller achieved an average text entry rate of 0.85 char/min tested with three healthy subjects.

Scherer et al. [36] proposed a three-class MI-based virtual keyboard to improve the spelling rate. They divided the interface into two horizontal parts: the upper part (20% area) was used to display selected letters, and the lower part consisted of a horizontal selection line and two vertical lines containing the letters in alphabetical order. In this design, 26 letters (A–Z) along with 2 special operations ("DELETE" and "OK") were considered, and 3 imaginary motor movements were used as control commands. This spelling paradigm achieved a mean performance of 1.99 char/min tested with three healthy subjects.

Blankertz et al. [7] proposed a two-class MI-based mental typewriter model called Hex-o-spell, where they mapped 30 target symbols with two control commands. Hex-o-spell was developed from the Hex system [41] designed for mobile devices. In this speller interface, letters were arranged alphabetically (along with four other symbols) and placed over six hexagons, each containing five symbols. This speller was controlled by two mental states for selecting a target in two selection stages. Imaginary right-hand movements were used to navigate clockwise among hexagons and imaginary right-foot movements were used to expand a pointed hexagon. After expanding, five characters from a selected hexagon will distribute among the first five hexagons in the second stage, and the sixth empty hexagon will be used to undo the last action performed. This work used an adaptive statistical data compression technique called prediction by partial matching as a **language model (LM)** to predict the next character based on previous characters. This speller was tested with two healthy subjects and achieved 2.3 char/min to 5 char/min for one subject and 4.6 char/min to 7.6 char/min for the other subject.

The Hex-o-spell idea has motivated many researchers to develop BCI spellers in the last two decades [34], and a number of variations of the Hex-o-spell were developed. MI-based BCI spellers like a predictive speller [11], asynchronous 2D cursor control speller [42], BrainTree [30], Oct-o-spell [8], and Spir-o-spell [27] were designed inspired by the Hex-o-spell.

D'Albis et al. [11] designed a synchronous operation mode-based predictive BCI speller. They proposed an original speller interface in which an LM was used to speed up character selection and provided word suggestions during the process. In this speller interface, letters were arranged alphabetically, along with "Space" equally divided into three groups. The fourth group included undo, prediction, and a dedicated menu with nine special operations. A target selection needed three stages in this interface and was powered by four MI movements. This speller achieved 3 char/min, 2.7 char/min, and 2 char/min when experimented with three healthy subjects, respectively.

A binary tree structured-based design, called BrainTree, was proposed by Perdikis et al. [30]. This speller paradigm provided options to the users to select any two MI classes from three different classes. The speller interface had a character bar with 26 letters, space, and backspace. A binary tree structure was followed in the selection of characters. The "Undo" command was incorporated in the interface and operated through user brisk movement, which was captured by monitoring a single bipolar **electromyography (EMG)** channel. In this work, the authors evaluated the performance of the speller with the target users and healthy participants. They achieved an overall performance of 1.47 char/min for motor-impaired and 1.84 char/min for healthy participants.

Another design level contribution for the BCI speller was made by Cao et al. [8] called as Oct-o-spell. This speller paradigm has a three-layer interface for character input, wherein the first layer, 26 alphabets, 10 digits, and 6 symbols ("Quit," "=," "←," "." "F1," and "Chn/Fn") were divided into 8 blocks of an octagon. In the second layer, there were eight different octagons to expand all characters of a selected block from the first layer. In the third layer, only two octagons were connected: One was "F1" with six punctuation marks, and the other was for "Quit." The character selection in this speller interface was carried out by moving the cursor through three MI classes. They achieved 9.66 char/min and 10.15 char/min mean performance for the non-predictive and predictive approaches, respectively.

Table 1. Summary of the Literature Survey on MI-Based BCI Spellers

Reference	#Tar.	#CC	LM	#Stg.	KLS	TSP	ECK	CPM	Remark
[28]	32	2	No	6	Static	Direct	Delete and undo	0.85	Hard to reach delete and undo operator. Did not follow any efficient symbol arrangement.
[36]	28	3	No	7	Static	Direct	Delete	1.99	Users have to perform 28 mental tasks in case of a missed target. There was no efficient error correction policy.
[7]	30	2	Yes	2	Dynamic	Direct	Delete and undo	4.87	Dynamic layout increases the average reaction time. Word/phrase-level error correction is difficult.
[11]	46	4	Yes	3	Dynamic	Direct	Delete char, delete word, and undo	2.57	The synchronous operation mode binds a task in a predefined time frame; hence, a user doesn't have full control on this speller.
[30]	28	3	Yes	NA	Static	Direct	Delete and undo	2.57	Undoing an action needs a separate control modality other than EEG. Here, error correction is difficult.
[8]	50	3	No	3	Static	By cursor movement	Delete and undo	9.66	Along with three distinct MI tasks, two simultaneous MI tasks are required to select few target characters, which leads to a complex selection process.
[44]	27	5	No	3	Static	Direct	Undo	6.67 (max)	A small set of symbols mapped with five control commands.

Table 1 summarizes the most popular MI-based BCI spellers from literature on different factors to understand the scope and motivation of this work. The factors considered in this table are number of targets (#Tar.) included, number of control commands (#CC) used, LM used for word prediction or not, number of selection stages (#Stg.) required for a target selection, **keyboard layout style (KLS)**, **target selection process (TSP)**, **error correction keys (ECK)** for error correction, and average entry rate (**character per minute (CPM)**).

Scope and Motivation. From the survey of the existing related literature, it was observed that the existing BCI spellers had a common issue in error correction (word-level and phrase-level) as they supported only sequences of backspaces. This correction policy was time-consuming and required more user effort. Hence, there is a scope to reduce users' efforts in error correction using an efficient error correction policy. Also, there is a scope to improve further the performance of a BCI speller following an efficient arrangement of the alphabet and other symbols necessary for text entry.

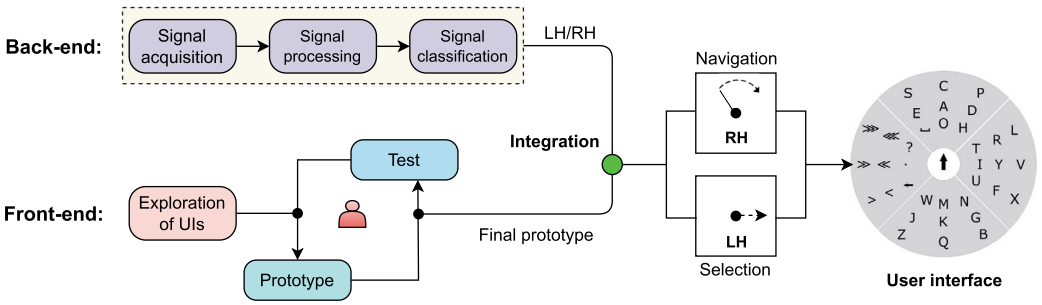


Fig. 1. Overview of the proposed text entry system development.

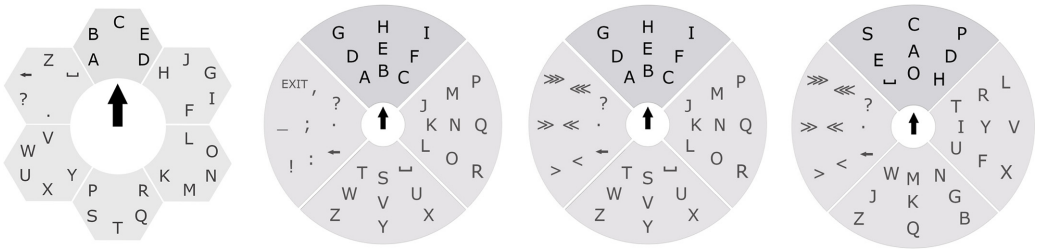
3 Proposed System

This section discusses the proposed MI-based text entry system consisting of three major parts: back-end, front-end, and the integration mechanism. Figure 1 gives an overview of the proposed approach. The back-end is a MI EEG signal classification pipeline to classify two distinct imagery motor movements (LH: left-hand and RH: right-hand). It contains modules such as signal acquisition, processing, and classification. On the other hand, the front-end is a graphical user interface developed through an iterative design and testing process after the initial design was explored. The final prototype with an efficient symbol arrangement was used as the user interface for entering texts. These two sub-systems are later integrated following an efficient integration mechanism to accomplish a complete text entry system, where the BCI sub-system collects EEG data, processes it, classifies it, and converts the classified output to the desired control commands for the front-end in real time. The user interface then fetches the command and acts accordingly. Suppose the user interface fetched the command as RH. In that case, it will rotate the pointer clockwise to navigate through the quarter circles, or if the command is LH, the interface will select and expand the pointed quarter circle. The following subsections describe the front-end design, the back-end system, and the integration of the front-end and back-end.

3.1 Front-End: The Speller Interface

In this work, a graphical user interface was finalized through an iterative design process starting with an initial design. Several modifications were considered based on user evaluations (discussed in Section 3.1.2). In the design space exploration process, the fourth prototype was found optimal while evaluating prototypes, which meets satisfactory results for our considered design rationale. Therefore, we considered up to Prototype-IV in the design space exploration process of this work. The first prototype (an initial design) was a non-predictive version of the Hex-o-spell [7] paradigm (see Figure 2(a)), and the other three were designs (see Figure 2(b)–(d)) proposed in this work to improve certain performance matrices. In Prototype-I, 30 target symbols were placed into 6 hexagons, and the other 3 prototypes included 36 target symbols placed into 4 quarters of a circular zone. Each quarter circle in each prototype contained nine characters in three groups, with three characters in each group.

The design of different prototypes and their evaluations are described in the following subsections. Section 3.1.1 describes the design space exploration which includes the exploration of UIs for choosing the initial design idea, rationales behind design modifications, and justification of each prototype design. Next, Section 3.1.2 includes the evaluation process for each prototype through five user experiments and selection of final prototype. Next, Section 3.1.3 illustrates the process



(a) Prototype-I [7]. (b) Prototype-II. (c) Prototype-III. (d) Prototype-IV.

Fig. 2. Four prototypes considered in design space exploration of this work, where the first prototype is the initial design based on the Hex-o-spell [7] and rest are novel designs.

of character selection on the final prototype (Prototype-IV). Lastly, Section 3.1.4 describes the handling of errors occurs during the text entry tasks for the final prototype.

3.1.1 Design Space Exploration. This section describes the iterative design and analyses of prototypes, and how the design rationales were formulated for design modifications. Note that Hex-o-spell [7] was chosen as the initial design idea. The literature survey revealed that Hex-o-spell is a popular MI-based BCI speller interface. This speller had influence as the baseline for many other works [8, 11, 27, 30, 42]. This proposed work also considered Hex-o-spell as the initial design idea and termed as Prototype-I. In Prototype-I (as shown in Figure 2(a)), six hexagons provide six selection options, and it accommodated 30 target symbols. In this prototype, for example, two tasks were required to reach the target "A," three tasks for the target "B," four tasks for the target "C," and so on. The total number of tasks is calculated as 195 for 30 symbols and the average task is 6.5 task per character. On the other hand, other three prototypes (see Figure 2(b)–(d)) were designed based on the novel "quarter circle" paradigm. The quarter-circle interfaces contained four quarter circles which needed four selection options. According to *Hick-Hyman law* [15, 17], the time a user takes to make a decision is proportional to the "*log*" of the number of options provided to them. Hence, it is reasonable to assume that reduced selection options (six hexagons to four quarter circles) will reduce the user's **reaction time (RT)**. In quarter-circle prototypes, a total of 36 target symbols were considered. Note that the average task in these prototypes was also calculated as 6.5 tasks per character. Hence, it was possible to accommodate six extra target symbols or characters than Prototype-I without increasing the average task. More detailed information about calculation of average tasks is discussed in Section 3.1.2. The following two rationales were behind the design of Prototype-II from Prototype-I.

Rationale-1. *To reduce the RT with a reduced number of selection options.*

Rationale-2. *To accommodate six extra target symbols without increase in the average task.*

The scope of further modification of Prototype-II arised from a user experiment. The experiment conducted on Prototype-I and Prototype-II revealed that correcting spelling errors at the word/phrase level was time-consuming and requires high effort from the user due to only one correction operator ("backspace") available on the speller interface. Hence, an efficient error correction method was needed to improve the overall performance, including entry-rate and EA. The rationale for designing the third prototype was formulated from this observation. The rationale for designing Prototype-III from Prototype-II is stated below.

Rationale-3. *Six text-cursor control keys are useful to improve the* ***correction efficiency (CE)*** *and reduces a user's effort in error correction at word/phrase level.*

To work with Rationale-3, six text-cursor navigation keys ("<," ">," "≪," "≫," "⋘," and "⋙") were introduced in Prototype-III. Note that, six redundant target symbols, "EXIT," "„" ":," ";," "!," and "_" in Prototype-II were replaced with the six text-cursor navigation keys mentioned above, and Prototype-III was designed. These six keys were planned to navigate the text cursor in the text field freely. The key "<" and ">" can move the text cursor one position (character-wise) left and right, respectively. The key "≪" and "≫" can move the text cursor word-wise left and right. The key "⋘" and "⋙" can move the cursor to the extreme-left and extreme-right, respectively.

Although EA and CE were improved in Prototype-III, as found from a user experiment, the entry-rate was not improved considerably. This observation from the experimental results of Prototype-III enlightened a way to enhance the performance of the speller with a special arrangement of symbols in the quarter circles, giving the idea of Prototype-IV. The rationale behind the design of Prototype-IV was as follows.

Rationale-4. *Arrangement of letters based on their frequency of occurrences in commonly spoken English words can improve the performance of a speller.*

In Prototype-IV, letters were placed in an order such that the most frequently used letters took a minimum effort to select. The frequencies of the English letters were calculated from the phrase set proposed by MacKenzie and Soukoreff [25] and listed in the decreasing order of the frequencies of occurrences (Letter:Frequency): E:1523, T:1080, O:1004, A:922, I:879, S:799, R:791, N:744, H:540, L:484, D:403, U:390, C:351, Y:303, M:296, G:276, P:250, F:248, W:235, B:178, V:154, K:141, J:35, X:34, Q:26, Z:13.

This letter-frequency order reminds a well-known data compression algorithm called Huffman coding [16], where the letter with highest frequency takes less number of bits for representation. The same idea is applicable in case of grouping the letters so that a frequently used letter should take a less number of tasks. Hence, the grouping and placement of the target symbols in this prototype were done based on the Huffman tree constructed using Huffman coding algorithm from 500 phrases [25] proposed by MacKenzie and Soukoreff. From this tree, the letters are traversed level-by-level (and right to left) and found the same letter-frequency order that was found from the phrase set. Figure 3(a) shows the level-wise arrangement of the letters from Huffman tree. Along with these letters, "space" which should be considered as another highest frequency target symbol used in any speller; hence, it was added on the top level (see Figure 3(b)). After that, for making groups of three letters, the last three rows were shifted towards right and created nine groups (see Figure 3(c)).

After grouping the letters, the next task was to place them on the speller interface. The proposed quarter circle speller interface provided spaces for 36 symbols (12 groups with 3 symbols in each group). The number of tasks to reach each position was determined and visualized in Figure 4(a) (left). Along with 26 letters and "space," 9 additional symbols (backspace, 2 punctuation marks, and 6 text-cursor navigation keys) were considered. For reducing the complexity of the letter arrangement and simplify the visual search for cursor navigation keys, these nine symbols were placed on the fourth quarter circle (see right part of Figure 4(a)) as was in Prototype-III. There are 27 positions (9 groups) left in the speller interface (see Figure 4(b)). In these nine groups of empty positions, the nine groups formed from the Huffman tree were placed as shown in Figure 4(c). From Figure 4(c) and the letter-frequency order, it was observed that placement of six letters ("H," "L," "G," "B," "V," and "X") violates the actual letter-frequency order, for example, letter "H" should take less

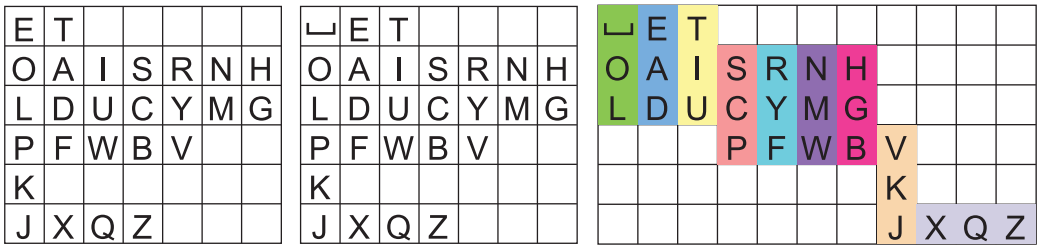


Fig. 3. Symbol arrangement and grouping: (a) level-wise arrangement of letters from Huffman tree, (b) adding "space" to the top level, and (c) shifting fourth, fifth, and sixth row towards right for making three-letters group.

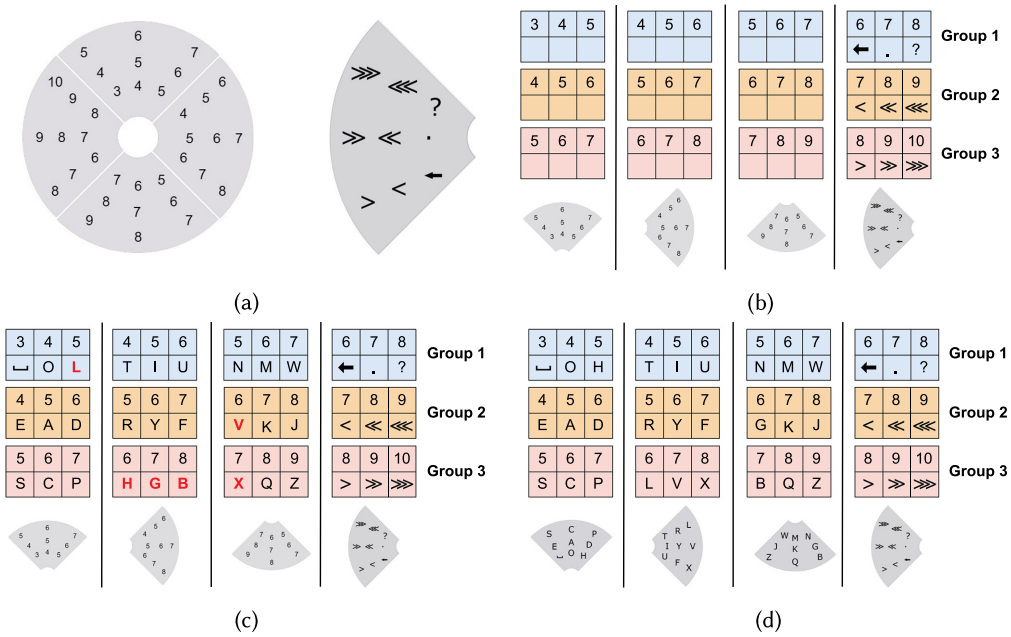


Fig. 4. Symbol placement on the speller interface: (a) the number of left-hand or right-hand tasks required to reach a particular position on the speller interface (left) and nine special symbols placed on the fourth quarter circle of the speller interface (right), (b) remaining 27 positions after placing special symbols on the speller interface, (c) placing English alphabet and space on the empty positions following the proposed grouping scheme, and (d) rearranging alphabet for correcting the latter-frequency order.

number of task than letter "L" but it's not as shown in Figure 4(c). To overcome this, the positions of "H" were swapped with "L," "G" with "V," and "B" with "X" and all nine groups were placed on the first three quarter circles of the speller interface (see Figure 4(d)) to construct the Prototype-IV. Again, a user experiment was conducted with the same set of participants to evaluate the prototype and validate the design rationale.

3.1.2 Evaluation of the Prototypes. This section discusses the user interface evaluation process. In this evaluation process, 5 experiments were conducted with 17 participants during the design and testing of each prototype. These experiments validate the design rationales considered in the design

Table 2. Specifications of the Personal Computer Used in This Work

Components	Units/Properties
CPU	Intel(R) Core i3-7100U @2.40 GHz
GPU	3.9 GB Intel(R) HD Graphics 620 and 2 GB NVIDIA GeForce 920MX
Memory	8 GB DDR3 RAM (primary) and 500 GB SSD (secondary)
Display	1,920 × 1,080 px, 75 Hz, 27 inches, FHD LED Anti-Glare and Anti-Flicker
Peripheral devices	Logitech MK270r USB Wireless (2.4 GHz) Keyboard and Mouse

space exploration process. The following paragraphs describe the participants' demographic information, the apparatus and text corpus used, and the performance evaluation matrices considered in this evaluation process.

Participants. The evaluations of user interfaces were carried out with 17 healthy volunteers (12 males and 5 females). The participants were students and faculty members with age ranges between 26 and 58 years ($Age_{\text{mean}} = 31.94$ and $Age_{\text{std}} = 9.63$). Participants performed text entry tasks with each prototype and their feedbacks were collected. Participant's identities were remained confidential during this evaluation process and referred to with unique IDs S01 to S17.

Apparatus. In this user interface evaluation process, a personal computer with the specification mentioned in Table 2 was used to run all the experiments. Additionally, keyboard and mouse were used during this prototype evaluation process.

Text Corpus. Two text phrases, "PACK MY BOX WITH FIVE DOZEN LIQUOR JUGS" and "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG," were used as the text corpus to evaluate the performance of each prototypes in Experiment-2, Experiment-3, and Experiment-4. These two phrases were chosen as they are pangram and contain all 26 English letters and the "space."

Performance Estimation Matrices. To evaluate the effectiveness of reduced selection options (from six to four), RT is the standard performance matrix used in this evaluation process. We used CPM as it is the well-known measure of entry rate for any text entry system. To measure the efficiency of the speller system and the usefulness of text-cursor navigation operators in error correction, we measured EA and CE. The performance metrics considered in this user evaluation process are defined below.

RT. The RT is the time required for visual search of a character plus the response time for the successful search. The average RT for 30 common symbols in Prototype-I and Prototype-II was measured using the equation given below:

$$RT_{\text{avg}} = \frac{1}{30} \sum_{i=1}^{30} RT(Symbol[i]). \tag{1}$$

EA. The term EA defines the percentage of presented text transferred correctly, and it can be measured using the formula given in Equation (2):

$$\text{EA} = \frac{|T| - INF}{|T|} \times 100\%. \tag{2}$$

In Equation (2), $|T|$ is the length of the text entered by a user, and INF is the number of incorrect characters/symbols not fixed.

CE. CE is defined as the ratio of the number of error corrections (i.e., number of incorrect fixed, IF) and keystrokes performed for correcting those errors (i.e., number of fixing symbols, F), as

shown in Equation (3):

$$\text{CE} = \frac{IF}{F}. \tag{3}$$

CPM. The number of character entered in 1 minute is measured as CPM. CPM can be measured in two different modes, error free CPM (CPM_{EF}) and CPM with typing errors (CPM_{WE}). These two performance matrices can be measured using Equations (4) and (5), respectively.

$$\text{CPM}_{\text{EF}} = \frac{|T| - 1}{\textit{Total time}} \times 60. \tag{4}$$

In Equation (4), **total time (TT)** (in seconds) is measured from the first key-press to the last key-press, and the constant 60 is the number of seconds per minute:

$$\text{CPM}_{\text{WE}} = \frac{|T| - 1}{\textit{Input time}} \times 60. \tag{5}$$

In Equations (4) and (5), IT is the time required to enter a text, excluding the error **correction time (CT)**. That is, $TT = IT + CT$.

Experiment-1: Validation of Rationale-1 and Rationale-2.

Objectives. The first objective of this experiment was to measure user's RT for visually searching a target character on Prototype-I and Prototype-II. Second objective was to validate the design Rationale-1 and Rationale-2.

Procedure. Experiment-1 was a task-based experiment, where a user had to visually search for a target character from a character group and respond if present (or not present) by pressing the "enter" (or "tab") key from the physical keyboard. The experiment was designed using the PsychoPy software [29], where a window with a target symbol on the left part of the screen and a group of symbols on the right was displayed. Participants must identify whether or not the symbol on the left part of the screen was present in the group. While the target symbol was not in the group, the user had to press the "Tab" key, and the system displays the next group (until the target character is found); otherwise, a user had to press the "Enter" key from the physical keyboard to complete the task. The experiment with each participant was conducted in three sessions with a 5-minute gap between two sessions. The time duration from the target character displayed on the left side of the screen for searching to the "enter" key pressed by the user was considered as the *RT* in this experiment and recorded for evaluation. Note that, the target symbols were chosen at random from 30 symbols exactly once for each session. The experiment was run with two designs: Prototype-I and Prototype-II counter-balanced. Further, only the symbols common in both Prototype-I and Prototype-II were considered in this experiment.

Considering the design Rationale-2, the task per character and the average task for Prototype-I and Prototype-II were calculated. Task per character means the number of left/right operations required to reach a particular character on a speller interface. Note that, in the calculation of task per character, users involvement were not required. The same had been calculated manually from the layout of Prototype-I and Prototype-II. Firstly, the sequence of left/right operations for each character was listed based on the character selection process of Prototype-I and Prototype-II. Then the number of task for each character was noted and the average task per character was calculated.

Results. As mentioned above, Experiment-1 was conducted in three sessions, and Table 3 shows the average RTs of 17 subjects or participants for 3 sessions. In each session, there were 30 symbols presented randomly, and the RT was calculated for a particular session using the formula shown in Equation (1).

Table 3. Mean RT of 17 Subjects for 3 Experimental Sessions with Prototype-I and Prototype-II

Prototypes	Session-1 (Seconds)	Session-2 (Seconds)	Session-3 (Seconds)	Mean ± SD
Prototype-I	3.97 ± 0.62	3.67 ± 0.53	3.15 ± 0.52	3.60 ± 0.48
Prototype-II	3.55 ± 0.54	3.06 ± 0.42	2.76 ± 0.46	3.13 ± 0.42

Table 4. Average Task per Target for Prototype-I and Prototype-II

Prototypes	30 Common Targets	Six Additional Targets	# Targets	# Task	Avg. Task
Prototype-I	A–Z ␣ ← . ?		30	195	6.5
Prototype-II	A–Z ␣ ← . ?	, _ : ; ! EXIT	36	234	6.5

Table 4 shows 30 common targets or characters for both the prototypes, 6 additional targets in Prototype-II, total number of targets (# target), total number of task (# task), and the average task per targets for Prototype-I and Prototype-II.

Conclusion. From Table 3, it was observed that RT for Prototype-II is lower than Prototype-I, which validates the design Rationale-1 (that is, reducing selection options reduces user's RT). For better justification a statistical test was performed to show the significant difference between Prototype-I and Prototype-II for the mean RT as dependent variable. The statistical measures are reported in the *Statistical Analysis* paragraph described below. From Table 4, it can be observed that extra six symbols could be accommodated in the proposed speller prototype without increasing the average task, which validates the design Rationale-2.

Experiment-2, -3, and -4: Performance Estimation of Prototypes.

Objectives. The objectives of the Experiment-2, -3, and -4 are mentioned below:

— *Experiment-2*: To measure the performance of Prototype-I and Prototype-II to justify the design modification.
— *Experiment-3*: To measure the performance of Prototype-III and compare with Prototype-II to validate the design Rationale-3.
— *Experiment-4*: To measure the performance of Prototype-IV and compare with Prototype-III to validate the Rationale-4.

Procedure. These three experiments followed the same procedure. In these experiments, each designed prototype includes two command buttons representing the functions of two mental states, and participants were instructed to control the spellers using those two buttons. The command button labeled "Right" was used to navigate the pointer clockwise, and the button labeled "Left" will expand a pointed character group. Note that no BCI inputs were considered for operating the speller interfaces in these experiments. Each participant was given two text phrases (mentioned above as "Text Corpus") in two sessions, and they performed the text entry operation for the given phrases. The task completion time, number of right and left operations performed, number of back operations and backspace used, number of text-cursor navigation operators used, and length of the transcribed text were recorded during the experiments.

Results. The performance of Prototype-I and Prototype-II was measured in Experiment-2. The Experiment-3 and Experiment-4 measure the performance of Prototype-III and Prototype-IV, respectively. The performance matrices [26] such as EA, CE, and CPM_{EF} were measured using

Table 5. Mean Performance of 17 Subjects for Each Prototypes

Prototype	Time (Seconds)	EA (%)	CE	CPM_{EF} (Char/Min)
Prototype-I	432.68 ± 39.99	87.42 ± 6.83	0.18 ± 0.08	5.47 ± 0.82
Prototype-II	405.47 ± 39.00	88.54 ± 7.14	0.15 ± 0.05	5.70 ± 0.94
Prototype-III	432.03 ± 63.78	97.09 ± 2.66	0.45 ± 0.11	5.84 ± 1.03
Prototype-IV	332.53 ± 56.76	97.15 ± 4.32	0.44 ± 0.14	7.30 ± 0.99

Equations (2)–(4), respectively. Results as the mean of 17 participants for the 3 experiments are shown in Table 5.

Conclusions. In Experiment-2, the performance of the first two prototypes was measured, and the experimental outcomes are shown in Table 5. Results highlight that the mean task completion time is less in Prototype-II (405.47 seconds) than in Prototype-I (432.68 seconds). Again, EA and CPM_{EF} show the superiority of Prototype-II over Prototype-I. These experimental results justified the reason for design modification, that is, Prototype-II from Prototype-I.

The performance of Prototype-III was measured in Experiment-3 and reported in Table 5. Table 5 shows that the EA and CE of Prototype-III (97.09% and 0.45) are higher than Prototype-II (88.54% and 0.15), which validates Rationale-3. The performance (entry rate) of Prototype-IV was remarkably improved from 5.84 char/min to 7.30 char/min (see Table 5) for rearranging symbols according to their frequency of use, which validates the design Rationale-4. The statistical measures also show the significant differences among the prototypes for EA, CE, and CPM_{EF}. The statistical measures are reported in the *Statistical Analysis* paragraph described below.

Experiment-5: Usability Assessment.

Objectives. This experiment was conducted to examine the advantages of text-cursor navigation keys included in Prototype-III and Prototype-IV for efficient error correction. Also, this experiment was to draw a comparison among four prototypes, where Prototype-I and Prototype-II without cursor navigation keys, whereas Prototype-III and Prototype-IV have the cursor navigation keys. Users feedback also collected through a post-experiment questionnaire for finding the optimal design. Note that, this experiment was performed after the design and descriptive analysis of all the prototypes.

Procedure. This experiment was conducted for all the designed prototypes with six erroneous phrases, and users were instructed to perform the correction tasks. This usability assessment process provides quantitative measures (task completion time, success rate, and number of keystrokes performed) and qualitative measures (subjective satisfaction by post-test questionnaire). The following paragraphs describe the selection of six phrases, embedding pre-determined errors, performing the error correction experiment, collecting user feedback, and analyzing experimental results.

(1) *Selection of phrases*: For this error correction task analysis experiment, 6 text phrases were selected from the 500 phrases proposed by MacKenzie and Soukoreff [25] which are shown below.
—WHAT YOU SEE IS WHAT YOU GET
—I DO NOT CARE IF YOU DO THAT
—WOULD YOU LIKE TO COME TO MY HOUSE

—IF YOU COME HOME LATE THE DOORS ARE LOCKED
—IF AT FIRST YOU DO NOT SUCCEED
—DID YOU HAVE A GOOD TIME

These phrases are selected following a proper investigation and analysis. First, the word frequency was calculated from the 500 phrases (*phrase_list*) and listed in sorted order. Second, the top 74 words with higher frequency (frequency ≥ 5) were considered and assigned to a *word_list*. Third, the *phrase_list* is sorted based on the higher number of words contained from the *word_list*. Fourth, the top six phrases were selected. The motivation behind choosing only six phrases is that there can be three types of error correction possible (insertion, deletion, and modification), and for analyzing the usefulness of cursor navigation keys, three errors were placed in three different positions (front, rear, and middle) of each phrase. Now, the permutation formula ($3P3$) gives six different combinations.

(2) *Embedding pre-determined errors*: In the above phrases, three errors were added in three different positions (front, middle, rear) of each phrase. The types of pre-determined errors are insertion (I), deletion (D), and modification (M). After embedding the errors, the phrases [errors] are as follows:

—**WHT** YOU SEE **ISS** WHAT YOU **GAT** [I D M]
—I **D** NOT CARE **IS** YOU DO **THAHT** [I M D]
—**WOULDE** YOU LIKE TO **CME** TO MY **HOUSS** [D I M]
—**IFF** YOU COME HOME **LETE** THE DOORS ARE **LOCKD** [D M I]
—IF **AS** FIRST YOU **D** NOT **SUCCEEED** [M I D]
—**DIS** YOU HAVE **AS** GOOD **TME** [M D I]

(3) *Conducting the experiment*: These six erroneous phrases were provided to participants in six sessions and observed the user behavior while performing correction tasks on all four designed prototypes. For the usability assessment, quantitative and qualitative measurements were used. The task completion time, number of keystrokes performed, and the success rate were measured.

(4) *Subjective assessment*: In addition, subjective assessments were done by the participants through a question-answer session with a set of questionnaires on usability [9, 13, 18] of the interfaces under test. Questionnaires used in this subjective assessment are mentioned in Table 6. Participants gave their feedback in five-point Likert scale (1: difficult, 2: moderately difficult, 3: average, 4: moderately easy, and 5: easy) and mentioned the problems they had faced while entering the phrases or correcting any misspelled character(s).

Results. From the correction-task experiment, user performance on each prototypes was measured and the average performance is reported in Table 7. Firstly, each subject's performance for six phrases was measured for individual prototypes, and then the average performance for each participant was calculated. Lastly, the average performance for each prototype was calculated and reported in Table 7. Note that, the errors made during the error correction tasks were taken into account.

The user feedback collected in the subjective assessment as a five-point Likert scale rating for each prototype was analyzed through statistical measures to identify the significant differences among the prototypes. Table 8 shows the average ratings of 17 participants for 4 prototypes for 4 questions (Q06, Q07, Q08, and Q09) related to error correction.

Conclusions. The outcome of the correction-task experiments revealed that the number of keystrokes performed were very low for the Prototype-IV followed byPrototype-III. Therefore, the

Table 6. Five-Point Likert Scale Post-Experiment Questionnaire Used in This Work for User Interface Satisfaction (QUIS) [9, 13, 18]

#	Questions	5-Point Rating
Q01	Reading characters on the screen	Difficult ... Easy
Q02	Visual searching for a character on the screen	Difficult ... Easy
Q03	Organization information on the screen	Confusing ... Clear
Q04	Sequence of scene	Confusing ... Clear
Q05	Use of colors	Poor ... Good
Q06	Undo an action	Difficult ... Easy
Q07	Insertion of a character	Difficult ... Easy
Q08	Deletion of a character	Difficult ... Easy
Q09	Modification of a character	Difficult ... Easy
Q10	System speed	Slow ... Fast
Q11	Learning to operate the system	Difficult ... Easy
Q12	Overall rating to the system	Difficult ... Easy

Table 7. Performance of Four Prototypes (Average of 6 Phrases, Then Average across 17 Participants) for the Correction-Task Experiment

Prototypes	Task Completion Time (Seconds)	Success Rate (%)	# Keystrokes	# Button Press
Prototype-I	447.23	92.10	59.00	397.17
Prototype-II	481.67	87.00	61.47	403.55
Prototype-III	183.00	100.00	23.91	167.41
Prototype-IV	157.33	98.43	18.50	124.25

Table 8. Mean Ratings of 17 Participants for Each Prototype for 4 Subjective Questions

Prototypes	Q06	Q07	Q08	Q09
Prototype-I	2.47 ± 0.80	1.76 ± 0.75	2.12 ± 0.70	2.12 ± 0.70
Prototype-II	3.29 ± 0.77	2.18 ± 0.81	1.94 ± 0.75	2.35 ± 0.86
Prototype-III	3.94 ± 0.90	3.71 ± 0.92	4.24 ± 0.75	4.47 ± 0.72
Prototype-IV	4.06 ± 0.90	4.00 ± 0.71	4.29 ± 0.77	4.47 ± 0.72

performance matrices from Table 7 infer that introducing the cursor navigation keys minimizes the user effort in error correction and increases the EA.

From the outcome of the subjective assessment (see Table 8) it was observed that Prototype-IV is having the highest average rating for the selected questions compare to other prototypes. Which indicates undoing an action, insertion, deletion, and modification of a character is comparatively easy in Prototype-IV than the other three prototypes.

Statistical Analysis: Statistical Significance among Prototypes.

Objectives. The objective of this statistical testing is to find the significant differences among the designed prototypes and selecting the optimal design.

Procedure. From the outcomes of the user experiments and user feedback, it was found that Prototype-IV was more suitable for integration with the back-end. Although the descriptive results show significant differences among the four prototypes, statistical testing is more prominent to validate our decision statement. The statistical test was performed in the following steps:

(1) A normality test was performed to determine the distribution of values of dependent variables. The normality test was performed for RT, speller performance, and for subjective assessment.

(2) Based on the outcomes of normality test, the appropriate significance test was identified.

(3) The parametric paired sample t-test was performed to identify the significance difference between Prototype-I and Prototype-II for the RT. The non-parametric version of the Repeated Measures ANOVA (Friedman) test was performed to find the significant differences among the four prototypes based on the speller performance (EA, CE, and CPM_{EF}) and subjective measures (Q06, Q07, Q08, and Q09).

(4) Following the significant Friedman test, *post-hoc* pairwise comparisons were conducted using the Wilcoxon signed-rank test with a Bonferroni correction applied to select the best prototype.

Statistical Tests and Results. Since we had a small sample size, determining the distribution of variable RT, speller performance, and Likert scale rank data of subjective measures was important for choosing an appropriate statistical method. A Shapiro-Wilk test was performed to determine the distribution of dependents variables. The Shapiro-Wilk test tests the null hypothesis that a sample $x_1, x_2, \ldots, x_n$ came from a normally distributed population or not based on the *test statistic (W)* and p-value. The *test statistic (W)* and p-value, W(34) = 0.981 and $p > 0.05$ ($p = 0.801$), retains the null hypothesis and shows that RT is normally distributed.

Again, the *test statistic (W)* and p-value showed a significant departure from the normality of the speller performance for EA ($W(68) = 0.888$, $p < 0.001$), for CE ($W(68) = 0.944$, $p = 0.004$), and for CPM_{EF} ($W(68) = 0.958$, $p = 0.021$). This normality test rejected the null hypothesis based on p-values, i.e., the data are not normally distributed for the above three dependent variables.

Lastly, the normality test on the subjective assessment data shows that these data are not normally distributed for Q06 ($W(68) = 0.905$, $p < 0.001$), for Q07 ($W(68) = 0.913$, $p < 0.001$), for Q08 ($W(68) = 0.895$, $p < 0.001$), and for Q09 ($W(68) = 0.879$, $p < 0.001$).

Based on the outcome of the normality tests, the parametric paired sample *t*-test was performed to check if there is any significant difference or not based on the RT before (pre-test with Prototype-I) and after (post-test with Prototype-II) design modifications. The test results show that there is a significant difference between the prototypes and RT is 0.468 seconds less when participants used Prototype-II, with a 95% confidence interval of 0.102 to 0.833 seconds. The *test statistics* and p-value are $t(16) = 2.71$ and $p = 0.015$, respectively.

To find the significant difference among the prototypes for speller performance matrices and subjective assessment Friedman test was performed. The Friedman test is a non-parametric version of Repeated Measures ANOVA test and compares outcome among more than two related groups. The Friedman test revealed a statistically significant difference among four prototypes for three dependent variables of speller performance, as follows:

—for EA, $\chi^2(3, N = 17) = 25.329, p < 0.001$,

—for CE, $\chi^2(3, N = 17) = 39.976, p < 0.001$, and

—for CPM_{EF}, $\chi^2(3, N = 17) = 15.635, p < 0.001$.

Table 9. Significant Differences between Two Prototypes for Pairwise Comparison Using Wilcoxon Signed-Rank (Bonferroni) *Post-Hoc* Test Performed on the Speller Performance Matrices and Subjective Measures

	Speller Performances			Subjective Measures			
Pairs of Comparison	EA	CE	CPM_{EF}	Q06	Q07	Q08	Q09
Prototype-I, Prototype-II	−1.12	0.03	−0.23	−0.82	−0.41	0.18	−0.24
Prototype-I, Prototype-III	−9.67*	−0.27*	−0.37	−1.47*	−1.94*	−2.12*	−2.35*
Prototype-I, Prototype-IV	−9.73*	−0.26*	−1.83*	−1.59*	−2.24*	−2.18*	−2.35*
Prototype-II, Prototype-III	−8.55*	−0.30*	−0.14	−0.65	−1.53*	−2.29*	−2.12*
Prototype-II, Prototype-IV	−8.61*	−0.29*	−1.60*	−0.77	−1.82*	−2.35*	−2.12*
Prototype-III, Prototype-IV	−0.06	−0.01	−1.46*	−0.12	−0.29	−0.06	0.00

Significant differences are denoted with asterisk (*) and mean difference (mean of $Prototype_i$ − mean of $Prototype_j$) shows the best prototype in each pair of comparison.

The Friedman test on the subjective assessment data also shows the significant differences among the prototypes for four subjective questions, as follows:

—for Q06, $\chi^2(3, N = 17) = 25.151, p < 0.001$,
—for Q07, $\chi^2(3, N = 17) = 39.745, p < 0.001$,
—for Q08, $\chi^2(3, N = 17) = 42.169, p < 0.001$, and
—for Q09, $\chi^2(3, N = 17) = 40.865, p < 0.001$.

The *post-hoc* test was conducted using the Wilcoxon signed-rank test with a Bonferroni correction applied, setting the significance level at $p < 0.0083$. The *post-hoc* test was conducted on speller performance matrices and four subjective questions with six possible pairs of comparisons. Table 9 shows the mean difference of two prototypes for the *post-hoc* test and the significant differences between the pairs for relevant dependent variables are indicated with asterisk (*) in the table.

Conclusions. The p-values for the significance test reject the null hypothesis and accept the alternative hypothesis. Hence, all four prototypes have significant differences for the considered dependent variables. The results of the text entry experiment with 17 participants for each prototype show EA, CE, and CPM_{EF} are higher in Prototype-IV ($Md = 98.45$, $Md = 0.43$, and $Md = 7.52$) in comparison to Prototype-I ($Md = 90.12$, $Md = 0.18$, and $Md = 5.27$), Prototype-II ($Md = 87.58$, $Md = 0.14$, and $Md = 5.48$), and Prototype-III ($Md = 97.47$, $Md = 0.46$, and $Md = 5.51$).

Again, the *post-hoc* study on speller performance and subjective assessment shows the pairwise significant differences of the designed prototypes. The negative value of the mean difference in Table 9 shows that the second prototype of each comparison pair performs better. Whereas the higher absolute value of the mean difference indicates the superiority of Prototype-IV and motivates to integrate with the back-end sub-system. Henceforth, the proposed BCI speller was constructed with Prototype-IV as the user interface and would be termed as QC Speller.

3.1.3 Character Selection Process. In the QC Speller interface (Prototype-IV), the selection of a character follows three selection stages, and the speller is powered by two mental states (imaginary left-hand/right-hand movements). The first quarter circle is always pointed in every stage, and users had to navigate the pointer to a target quarter circle. The imaginary right-hand movements are used to navigate the pointer clockwise, and the imaginary left-hand movements select a pointed quarter circle. The three-stage character selection procedure is illustrated in Figure 5 where the word "HELP" is considered as an example and Figure 5 shows the selection process of the character "P."

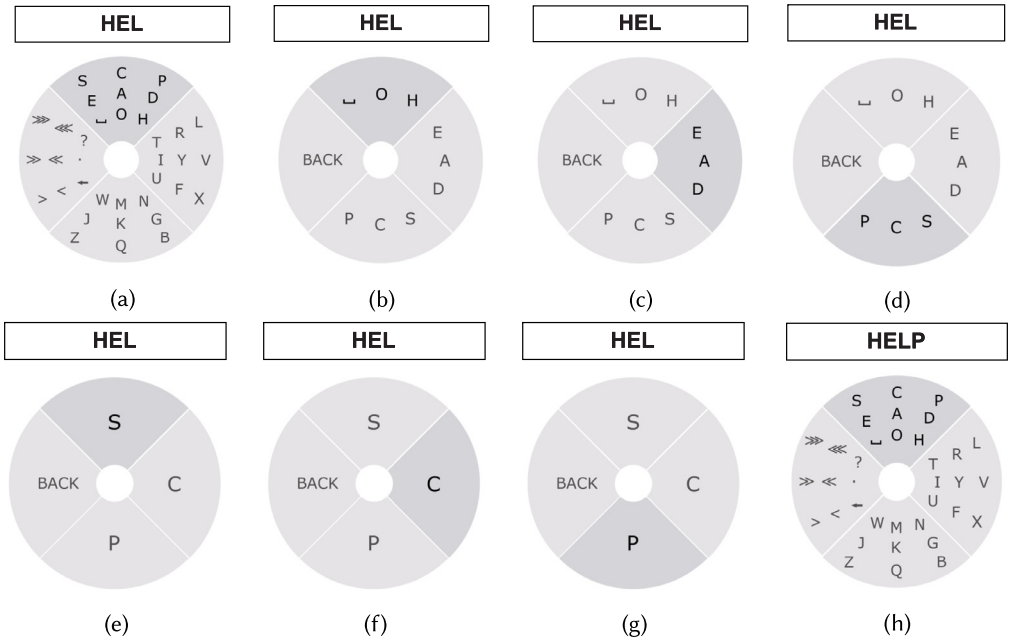


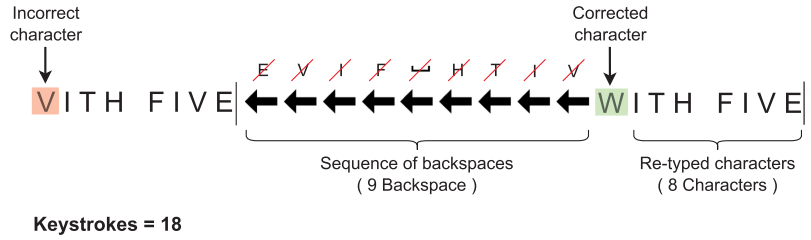
Fig. 5. An example of selecting a target character (say "P") on the QC Speller interface (Prototype-IV).

Figure 5(a) shows the first stage of selection. Figure 5(b)–(d) shows the second stage of selection. Figure 5(e)–(g) shows the third stage from where the target character is selected and returned to the initial layout (see Figure 5(h)). For the above example, a user must type "P" to complete the word "HELP." The target character "P" is present in the first quarter circle, and the pointer is currently pointed at the same. Users have to perform imaginary left-hand gestures to select and expand the characters from the first quarter circle. Upon selection of the pointed character group, the layout of the speller will change (see Figure 5(b)). Now first three quarter circles contain nine characters (three characters in each) and fourth quarter circle contains the "BACK" operator for returning to the previous stage. Now, the target character "P" is present on the third quarter circle. Users have to perform two continuous imaginary right-hand gesture to reach the third quarter circle and one left-hand MI gesture to select it. This process will continue until the target character gets typed, and the speller returns to its initial key layout after selecting the target character.

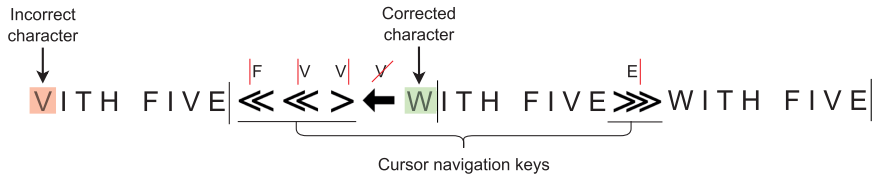
3.1.4 Error Handling. A user may encounter two types of errors in the proposed QC Speller. The first type of error will occur for selecting a wrong character group, and the second for typing a non-target character. The "BACK" function is included in this speller paradigm to undo an action and return to its previous stage to handle the first type of error. A user can correct the second type of error by deleting the misspelled character using the backspace ("←") key and with the help of text-cursor navigation keys.

Text-cursor navigation keys are found helpful in word/phrase level error correction as they reduce the number of keystrokes (backspaces). The following subsection will describe the usefulness of text-cursor navigation keys included in the proposed QC Speller.

A correction task can be one of three categories: insertion, deletion, and modification. User efforts for a correction task can be defined as the number of keystrokes performed during the correction task. For any standard BCI speller paradigm (existing), the number of keystrokes in word/phrase level error correction will be high for only one correction operator (backspace) option.



(a) Correction using sequence of backspaces.



(b) Correction using cursor navigation keys.

Fig. 6. An example of error correction task with and without cursor navigation keys.

A user has to perform a sequence of backspaces until they reach and erase the incorrect character and the user has to retype all the characters erased.

For example, consider a scenario from the user interface evaluation process where a subject (S02) typed the word "VITH" instead of the word "WITH" using Prototype-II for the presented text "PACK MY BOX WITH FIVE DOZEN LIQUOR JUGS" and noticed it after completing up to "PACK MY BOX VITH FIVE." The user used a sequence of backspaces for the modification task, corrected "V" as "W," and retyped "ITH FIVE" as shown in Figure 6(a). Here, the user performed nine backspaces to reach the incorrect character by erasing the characters followed by the incorrect character and retyped all the characters along with the modified one. Note that, in this correction process, a user may misspell a letter and needs another correction effort which is not considered in this modification scenario. The number of keystrokes for the modification operation is 18 (see Figure 6(a)). Since the average task for a keystroke in Prototype-II is 6.5 (which means, on average 6.5 MI tasks are required to reach one symbol and select it), therefore the user needs 117 (18×6.5) tasks in such a modification operation and it is indeed a tough job.

The number of tasks can be reduced using the text-cursor navigation key introduced in Prototype-III and Prototype-IV. Consider Figure 6(b), where the optimal solution for the above-specified modification task is illustrated. Here, three keystrokes (three text-cursor navigation operators) are used to reach the incorrect character, and one backspace operator is used to erase the incorrect character. After typing the correct character, a text-cursor navigation operator moved the cursor at the end of the input stream to continue typing. In this scenario, the number of keystrokes needed is 6. The overall tasks a user need to perform in this modification operation is 39 (6×6.5), which is considerably less compared to the tasks in Prototype-II.

3.2 Back-End: The BCI Sub-System

The proposed QC Speller uses two mental states of imaginary, right-hand and left-hand, movements as BCI inputs. These BCI inputs must be distinctly identified by a classification model and converted

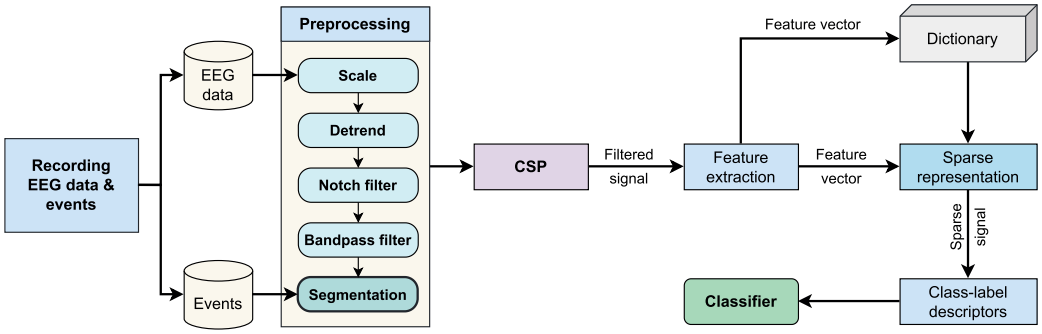


Fig. 7. Overview of the construction of SRC.

into control commands to operate the speller interface. This work adopted the sparsity-based classification model proposed by Sreeja et al. [37] for the BCI sub-system. The following are the rationales behind selecting the **sparse representation-based classifier (SRC)**: (1) SRC gives better accuracy compare to the traditional classification models, (2) SRC is faster, and (3) SRC overcomes session and subject variability of EEG data [24, 37, 38]. This work followed the same procedure (as in [37]) to build the classification model on two-class MI EEG data. Figure 7 shows an overview of the BCI sub-system. This approach first used a **common spatial pattern (CSP)** to filter the MI signals spatially. An over-complete dictionary matrix was constructed using statistical features, frequency band power, **discrete cosine transform (DCT)**, and wavelet features. Finally, an SRC model was built for distinguishing imaginary right-hand and left-hand movements.

MI EEG signals were recorded for designing the back-end, and the BCI sub-system and subject-specific classification models were created. The construction process of the back-end system is briefly described below.

Recording EEG Data and Events. Left-hand and right-hand MI EEG signals were recorded using an 8-channel OpenBCI Cyton board equipped with ThinkPulse Active dry electrodes and stored in text files. The stimulus for the data recording experiment was shown using a graphical user interface, and events were also recorded in text files.

Pre-Processing. The following pre-processing steps were performed on the raw EEG data for cleaning and segmenting data for further processing:

- *Scaling*: Scaling was done to convert EEG data from Cyton *int32* format to μV format by multiplying with a scale factor.
- *Detrending*: Scaled data were detrended using **common average reference (CAR)**, where mean of scaled data was subtracted from each data points.
- *Notch filter*: A 50 Hz notch filter was applied to the data for removing the power-line noise.
- *Bandpass filter*: For removing redundant higher and lower frequency data a 1–40 Hz bandpass filter was applied.
- *Segmentation*: After applying the temporal filters EEG data were first segmented based on the events and then based on time.

CSP. The CSP was applied to the segmented data to increase the variance of one class and decrease the variance of another class. CSP operates by analyzing the covariance between EEG channels and constructing spatial filters that are tailored to discriminate between different brain states. These filters are designed to extract the most discriminating spatial patterns from the EEG data, which can then be used as features for classification or further analysis.

Feature Extraction. From the spatially filtered data, seven statistical features (mean, median, SD, skewness, kurtosis, maximum, and minimum), seven time-domain features (Hjorth ability, Hjorth mobility, Hjorth complexity, 1st difference mean and maximum, and 2nd difference mean and maximum), two frequency band powers (alpha and beta band power), DCT coefficient, and discrete wavelet features (approximation coefficient energy and detail coefficient energy) were extracted. The extracted feature vector was a higher-dimension vector that consumed more computational power and processing time. In building a real-time system, the lower processing time is highly considerable. Hence, redundant features were eliminated from the extracted feature vector to reduce processing time in the live system. From the 19 extracted features, 9 relevant features for each channel were selected using the random forest classifier based on feature importance, and a feature vector was constructed. Later, the feature vector was divided into train and test sets.

Dictionary Construction and Sparse Representation. A over-complete dictionary was constructed from the training set. Given a feature vector from a test sample, express this vector as a sparse linear combination of the atoms (columns) in the dictionary. This step involves solving an optimization problem to find the sparse coefficients that best represent the test sample using the dictionary. The sparse representation of the feature vector was constructed using the **orthogonal matching pursuit (OMP)** algorithm [37].

Classification. The sparse representation of the test sample over the dictionary constructed in training was used for classification. The class label can be determined based on the similarity or reconstruction error between the test sample and its sparse representation using the training dictionary. This sparse matrix uses a class-level descriptor and acts as a classifier [37].

3.3 Integration of Front-End and Back-End Sub-Systems

The front-end sub-system was then integrated with the back-end sub-system to develop the fully functional QC Speller as a close-loop system. Figure 8 shows the overview of the closed-loop integrated mechanism of the live system. The integrated close-loop system consisted of two parallel threads: one managing data acquisition, processing, and interpreting the signal received (back-end) and the other handling the user interface with the custom keyboard layout (front-end). Communication between the two threads was dealt with through a shared memory, which handles the timestamp of EEG data points collected in real time and the class label (classifier output) associated with the data received. When the prediction confidence goes past a certain threshold, *Thread-1* (which handles the back-end) writes the timestamp of the prediction and the predicted class label on the shared block of memory by acquiring a mutex lock first. *Thread-2* (which handles front-end) reads timestamp from the shared memory and when it detects a change in the timestamp in the shared memory, it reads the prediction label from the shared memory block and changes the UI according to the prediction. The communication between the threads and the complete text entry process for the speller system is shown as a pseudocode logic in Algorithms 1 to 3.

Signal Acquisition. EEG data were collected in real time using the OpenBCI Cyton board and ThinkPulse active EEG sensors at 250 Hz sampling rate through OpenBCI GUI. Note that, both front-end and back-end should be synced with a fixed rate. In this implementation it was fixed at a refresh rate of 250 Hz.

Signal Transfer. Data received from the Cyton board through the OpenBCI GUI was transmitted locally using the **lab streaming layer (LSL)** to the BCI sub-system for processing. The LSL system enables real-time synchronization and streaming of diverse data sources (e.g., EEG, ECG) for seamless integration in scientific research. It provides a standard protocol facilitating communication between different devices and software tools.

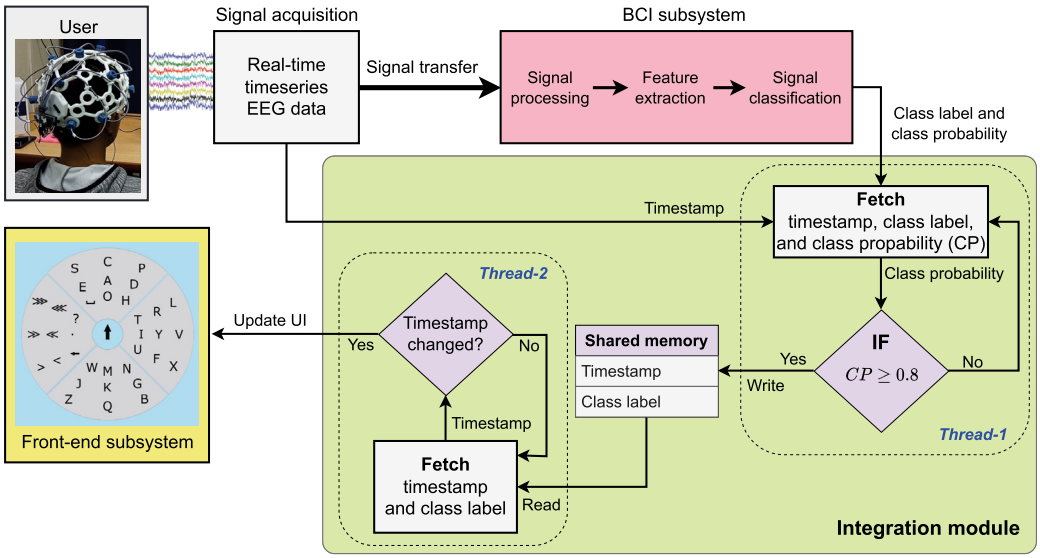


Fig. 8. The closed-loop integration of the front-end and back-end sub-system to build the real-time text entry system.

Algorithm 1: The Main Function for Creating and Starting Threads

Global: *predict_time* ← NULL, *last_predict_time* ← NULL, *pred* ← NULL, *LOCK* ← CREATE_MUTEX_LOCK()

```
Function MAIN()
    try:
        thread1 ← CREATE_THREAD(GUI_handler) // For handling front-end
        thread2 ← CREATE_THREAD(prediction_handler) // For handling back-end
        MAKE_THREAD_DAEMON(thread1, thread2)
        START_THREAD(thread1, thread2)
        JOIN_THREAD(thread1, thread2)
    catch:
        Output: Error in thread pool
    finally:
        DESTROY_THREAD(thread) // Destroy thread
    end
end
```

Signal Processing. The BCI sub-system collects the data transferred by LSL until a buffer of 250×8 data points is filled, considered 1 epoch. Here, a 250×8 data points buffer is considered as the refresh rate of the front-end and back-end is 250 Hz, and the number of channels is 8. Then, the collected epoch goes through four levels of pre-processing: scaling, detrending, notch filter, and bandpass filter. Finally, the CSP was applied.

Feature Extraction. The statistical features (mean, SD, skewness, and kurtosis), frequency band power (alpha and beta band power), DCT coefficient, and discrete wavelet features (approximation

Algorithm 2: Function for Creating and Updating Speller Interface Based on the Prediction Output Provided by the Back-End

```
Function GUI_handler()
    app ← CREATE_APPLICATION() // Create GUI for application and start application
    window ← CREATE_WINDOW(app)
    while window_is_open do
        UPDATE_GUI(window) // Update GUI after every 1-second delay
    end
end
Function UPDATE_GUI(window)
    LOCK(LOCK){
    if predict_time != last_predict_time then
        last_predict_time ← predict_time
        if pred == 'Left' then
            window.SELECTION()
        else
            window.NAVIGATION()
        end
    end
    } UNLOCK(LOCK)
end
```

coefficient energy and detail coefficient energy) were extracted from the processed EEG epoch acquired in 1 second. A total of 72 features (9 for each channel) were extracted, and a single feature vector of 1×72 dimension was constructed. The transpose of the feature vector (72×1) was passed to the classifier for predicting the desired class.

Classification. The prediction of the intended command (left-hand or right-hand command) was made by passing the sparse representation of the 72×1 feature vector to the SRC. When the probability of the predicted class being accurate exceeds a certain threshold, it writes the timestamp of the most recent data stream input and the predicted class into the shared memory.

Updating User Interface. The GUI refreshes or updates at 250 Hz and reads the contents of the shared memory for any change in timestamp. If a timestamp change is noticed, it reads the prediction stored in the shared memory block (right/left) and makes the necessary changes to the UI.

4 Experiment with QC Speller

After integration of the back-end with the front-end two experiments were conducted to create classification model for each subject and to validate the working of the proposed QC Speller. The objectives of the experiment, demographic information of the participants, and the apparatus used in the experiments are discussed in the following subsections.

4.1 Objective

The objectives of the experiments were as follows:

— *Experiment-A*: Collecting EEG data and building subject-specific classification models.
— *Experiment-B*: Performing text entry tasks in real time to evaluate and validate the proposed text entry system.

Algorithm 3: Function for Handling Back-End Sub-System and Providing Control Commands to the Front-End of the Speller System in Real-Time

```
Function prediction_handler()
    EEG_stream ← CREATE_EEG_STREAM() // Create EEG stream
    prediction_model ← LOAD_PREDICTION_MODEL() // Load the prediction model
    buffer ← ARRAY(NULL) // Initialize empty buffer
    buffer_size ← 250, threshold ← 0.8
    while EEG_stream_is_open do
        sample ← GET_EEG_SAMPLE(EEG_stream) // Get EEG sample
        time ← GET_CURRENT_TIME() // Get current time
        if sample is NOT NULL then
            buffer ← APPEND(buffer, sample)
        else
            CONTINUE
        end
        if LENGTH(buffer) > buffer_size then
            buffer ← SLICE(buffer, 1, buffer_size) // Remove oldest sample if buffer is
                overflowing
        else
            CONTINUE
        end
        feature_vector ← EXTRACT_FEATURES(buffer, nine_features) // Extract nine features
            and predict class
        prediction_class, prediction_probability ← PREDICT(prediction_model, feature_vector)
        if MAX(prediction_probability) > threshold then
            LOCK(LOCK){
            if prediction_class == 0 then
                pred ← "Left"
            else
                pred ← "Right"
            end
            predict_time ← time
            } UNLOCK(LOCK)
        end
    end
end
```

4.2 Participants

EEG signals were recorded from 10 subjects participated in the experiments. The subjects are participated from the Indian Institute of Cerebral Palsy (www.iicpindia.org) and NIEPMD, India (www.niepmdexaminationsnber.com). A brief profile of each subject is presented in Table 10, where each subject is referred to with a unique ID such as Subject-A, Subject-B, up to Subject-J. Their ages were between 24 and 34 (AGE_{avg} = 29.30 and AGE_{stdev} = 3.37). None of them had any BCI knowledge and did not participate in any BCI experiment. Participants were informed about the experimental protocol and objectives before experimenting. The Institute Ethical Committee, IIT Kharagpur, provided the ethical clearance for conducting experiments on humans.

Table 10. Demographic Information of Subjects Participated in EEG Data Acquisition and Real-Time Text Entry Experiments

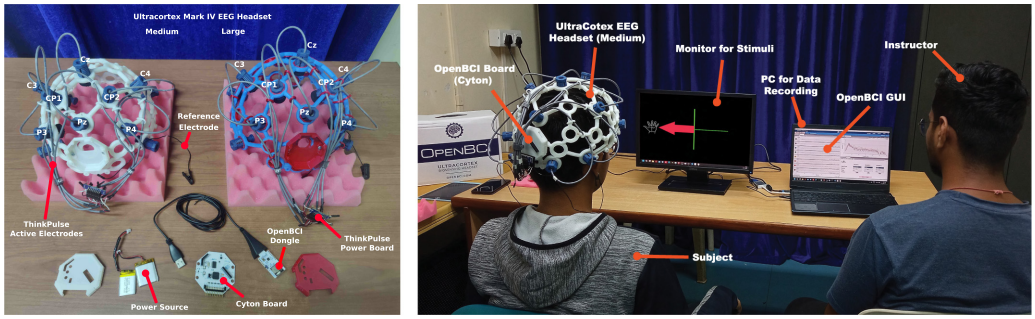
Subject	Sex	Age	Diagnosis	Disability on	Category	Sub-Category
Subject-A	Female	34	C6	Below neck	Spinal cord injury	Complete
Subject-B	Male	28	Limb-Girdle	Shoulders	Muscular dystrophy	Mild
Subject-C	Male	27	Myotonic	Both hand	Muscular dystrophy	Mild
Subject-D	Female	29	Hemiplegia	Right side	Cerebral palsy	Severe
Subject-E	Male	24	Hemiplegia	Right side	Cerebral palsy	Severe
Subject-F	Male	31	Hemiplegia	Left side	Cerebral palsy	Severe
Subject-G	Female	28	Diplegia	Both arms	Cerebral palsy	Severe
Subject-H	Male	26	Diplegia	Both legs	Cerebral palsy	Severe
Subject-I	Male	34	Diplegia	Both legs	Cerebral palsy	Severe
Subject-J	Male	32	Monoplegia	Left arm	Cerebral palsy	Mild

4.3 Apparatus

4.3.1 Hardware Setup. EEG signals were collected using an 8-channel OpenBCI Cyton biosensing board and Ultracortex Mark IV 3D printed EEG cap assembled with ThinkPulse (flexible polymeric dry sensors) dry active electrodes. The Ultracortex Mark IV EEG headset provides 35 positions for placing electrodes based on the region of interest, and eight electrodes were placed on the headset. The eight electrodes are placed over the central sulcus (C3, Cz, and C4), primary somatosensory (CP1 and CP2), and parietal lobe (P3, Pz, and P4) following the 10–20 system of EEG sensor placement. One reference electrode (A2) was placed on the right earlobe connected to the SRB2 pin on the Cyton board. No BIAS or GND electrodes were used as suggested in the OpenBCI documentation for ThinkPulse active electrodes setup. The sampling frequency was set to 250 Hz as an 8-channel Cyton board providing a maximum sampling frequency of 250 Hz. Before recording the data, it was ensured that the gain was set to 8 or less (we set gain as 4) and bias-include as NO (suggested for ThinkPulse active electrodes). Also, it was ensured that the impedance for each channel was less than 5 $K\Omega$. As EEG data were more noise-sensitive, the data were recorded in a controlled environment. Glass walls surrounded the recording room to prevent external noise, and the subject was seated on a foam-coated, non-movable plastic chair with feet resting on a footrest and arms resting on a wooden table. Before starting the experiment, the temperature of the recording room was maintained according to the participants' comfort through an **air conditioner (AC)**. Although, the AC was turned OFF during the experiment.

EEG data captured by the Cyton board were wirelessly transferred by the OpenBCI RFduino BLE radio module (dongle) to the recording PC for storage. PC as used in earlier experiments (specification given in Table 2) was used to collect EEG data for building classifier and to run the text entry in real time. The CPU and GPU specification, primary memory, and SSD disk space were supported enough to run OpenBCI GUI, speller interface, and several Python scripts without interruption. An additional 27-inch Anti-Glare, Anti-Flicker full HD LED monitor was used to display stimulus for data collection and display speller interface for text entry. The hardware components used for EEG data recording are shown in Figure 9(a).

4.3.2 Software Components. OpenBCI GUI was used to control the acquisition of EEG data and visualize data in real time. OpenBCI GUI provides real-time data-sharing protocols for sharing EEG data with external applications. The LSL protocol through a Python library called pylsl was used to transfer EEG data from the Cyton board to the BCI pipeline for real-time processing



(a) (b)

Fig. 9. (a) OpenBCI hardware components used in this work. (b) Recording session for a subject.

Table 11. Details of the Text Corpus Used in the Text Entry Experiment

Phrases in the Text Corpus	Properties of the Text Corpus
Phrase 1: win first prize in the contest Phrase 2: a little encouragement is needed Phrase 3: I agree with you Phrase 4: video camera with a zoom lens Phrase 5: the quick brown fox jumped	Minimum phrase length was 16, maximum phrase length is 32, average phrase length was 26.6, number of words is 26, number of unique words was 23, minimum word length was 1, maximum word length was 13, average word length was 4.86, number of letters was 112, number of spaces was 21, and correlation with English was 100%.

and classification. Several signal processing and machine learning Python packages were used in this work. The PyQt5 and Qt Designer were used to design the graphical user interface of the proposed speller.

4.3.3 Text Corpus. In this work, five phrases from a standard phrase set [25] proposed by MacKenzie and Soukoreff were considered as a text corpus. These phrases were chosen with maintaining properties of the complete text corpus [25]. The detailed information of the text corpus is given in Table 11.

4.4 Experiment-A

The main objective of this experiment was to collect left-hand and right-hand MI EEG data from the participants and building classification models. This experiment follows a standard protocol for data acquisition and processing. The complete process of building subject-specific classification models is discussed in the following subsections.

4.4.1 Data Acquisition. Left-hand and right-hand MI EEG data were collected from 10 subjects using OpenBCI Biosensing Board (Cyton) over 4 sessions for each subject. Each session lasted almost 25 minutes, including recording and preparation time. At the beginning of each session, there was a 20-second baseline duration for recording open-eye and eye-closed EEG data. After the baseline part, a beep for 1 second was performed to alert the subject to prepare for the trials. Each recording session includes 25 trials of imaginary left-hand movements and another 25 trials of imaginary right-hand actions. For each MI movements visual cue was shown for 6 seconds but the trial was considered as 5 seconds (last 5 second of the 6 seconds visual cue) for later processing.

During the trials, participants were instructed not to blink their eyes and move body parts. Before each trial started, a 4-second fixation gap was shown to get user attention on the trials and provide a resting period for blinking eyes.

The sampling frequency of the recording data was set as 250 Hz, and OpenBCI GUI stored each sample along with unique timestamps in a text file. The visual stimulus for recording data was displayed to the subjects using a GUI designed in Python using the PyQt5 library. The events were stored in a text file along with the timestamps. Figure 9(b) shows the experimental setup and ongoing recording process for a particular session of a subject.

4.4.2 Signal Processing and Feature Extraction. The EEG values parsed from the Cyton board's binary stream were $int32$ numbers (termed "counts"). Individual channel data were multiplied with the scale factor to get the values in proper form (in μV). Equation (6) was used to calculate the scale factor as mentioned in the OpenBCI Cyton documentation:

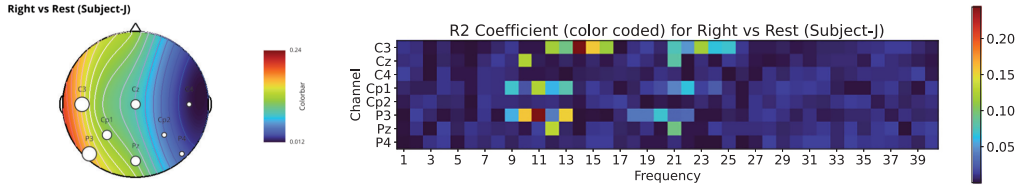
$$ScaleFactor\ (\mu V/count) = \frac{4.5 \times 10^6}{gain \times (2^{23} - 1)}. \tag{6}$$

The pre-processing was done on each session data of individual subjects. As the gain of the Cyton board was set as 4, the exact value of the scale factor was calculated as 0.13411 and multiplied with the data to get the proper μV format. After scaling, the data were detrended for each session data using CAR, and temporal filters were applied. A 50 Hz notch filter was used to remove the power-line noise, and a 1–40 Hz bandpass filter was applied to remove the redundant lower and higher frequency data. The EEG data for the baseline, audio stimulus, fixation duration, and the first second from each trial were removed based on the events. The segmented data for each session included 50 trials (25 trials for left-hand and 25 trials for right-hand) of 5-second epochs for 8 channels. The shape of the filtered data is $50 \times 1{,}250 \times 8$ for each session. Later, each 5-second epoch was segmented into five 1-second sub-epochs (the shape becomes $250 \times 250 \times 8$), and four sessions were merged (the total shape becomes $1{,}000 \times 250 \times 8$).

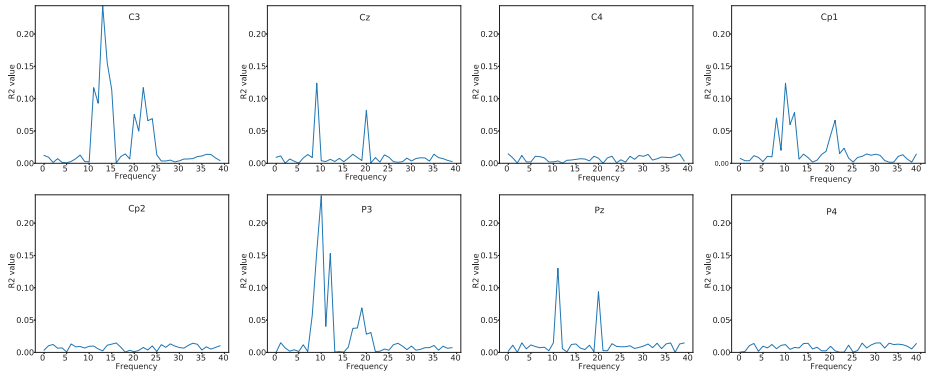
For verifying that the collected EEG data actually represent a MI task or not, the spectral power was estimated. The spectral power of the EEG signal was estimated for right/left MI tasks over the rest data, which was formulated from the fixation duration EEG data (with no MI tasks). Figure 10 visualizes the results of spectral estimation for right versus rest EEG data of Subject-J (selected as random, for example). The r^2 coefficient was calculated and pictured in Figure 10(a) as topographical map for the frequency (14 Hz) with maximum r^2 value. The color-coded heatmap for the r^2 coefficient (a function of frequency versus channel) is visualized in Figure 10(b) and in Figure 10(c) as line plots for each channel. The r^2 coefficient values represent the quality of EEG data for MI tasks with respect to the rest. Topomap shows the activeness of the left-hemisphere for right-hand MI signal and the heatmap shows the active frequency band (8–30 Hz) for the MI signals.

To enhance the signal localization and increase the discrimination properties of two-class data, a spatial filter called CSP [23, 37] was applied for each epoch of each class data. The spatially filtered data X_1^{CSP} and X_2^{CSP} obtained for two different classes were passed to the next step for feature extraction.

From the spatially filtered data, for each class, statistical features (mean, SD, skewness, and kurtosis), frequency band power (alpha and beta band), DCT coefficient, and discrete wavelet features (approximation coefficient energy and detail coefficient energy) were extracted for each channel data. Altogether, 72 features were extracted and merged in a single feature vector F_1 for one class and F_2 for another class. An over-complete dictionary (D) was constructed by concatenating the transpose of these two feature vectors (F_1^T and F_2^T), where each atom (column) of the dictionary represents a feature vector for a trial signal [37]. The shape of the dictionary was $72 \times 1{,}000$, where



(a) Topomap for r^2 value (max) (b) Coefficient r^2 as a function of frequency and channel (color coded).



(c) r^2 coefficient of eight channels.

Fig. 10. The r^2 coefficient as a function of channel and frequency for the EEG data of Subject-J for the "Right-hand versus Rest condition."

72 was the length of feature vector and 1,000 was the number of trials combined for two classes. Later, a sparse representation (S) of the dictionary matrix was formed using the OMP greedy algorithm. The results of this sparse representation were used for classification by applying a suitable classification rule (in this case, it was "variance between two class data"). To overcome the subject variability of EEG data, the classification model was created separately for each subject.

4.4.3 Sparse Representation, Classification, and Model Validation. Data were divided into training and testing sets using a 10-fold cross-validation scheme, and an SRC was created from the training data. Classification models were evaluated using the testing data for each subject. In this evaluation process, **classification accuracy (CA)**, precision, recall, and kappa coefficient (K) matrices were calculated and represented in Table 12.

Although, there are several classification techniques [40] reported in the literature for classification of EEG signal with considerable accuracy, it was found that sparse representation classifier is more suitable for real-time applications [37]. Different machine learning classification models were created for the training data and performance was measured. The classification performances shows proposed sparse representation classifier outperforms other classification models. The performance of the sparse representation classifier incorporated in this work is also considerable for constructing a real-time speller system.

4.5 Experiment-B

The general procedure and experimental setups used in the recording sessions were also followed in the real-time text entry part. Participants were seated in a chair with extended arms and legs during the text entry tasks. Their arms were resting on the desk, and their legs were resting on the footrest. A display (specification given in Table 2) was placed in front of participants at a distance between 0.5 and 0.75 m with a vertical viewing angle of approximately 10–15 degrees [12]. The

Table 12. The Mean (10-Fold) Offline Performance of Sparse Representation Classifier for Each Subject

Subject	CA (%)	Precision	Recall	K
Subject-A	95.50	0.92	0.95	0.87
Subject-B	96.92	0.99	0.95	0.94
Subject-C	81.00	0.82	0.81	0.62
Subject-D	85.79	0.90	0.83	0.72
Subject-E	90.00	0.93	0.90	0.80
Subject-F	85.50	0.86	0.85	0.71
Subject-G	91.04	0.86	0.96	0.83
Subject-H	94.97	0.97	0.93	0.90
Subject-I	83.00	0.83	0.83	0.66
Subject-J	92.50	0.93	0.92	0.85
Mean	89.62 ± 5.54	0.90 ± 0.06	0.89 ± 0.06	0.79 ± 0.11

average time taken for setting up the experiment and demonstrating the working principle of the speller to users was around 20 minutes. After setting up, users were provided with the running application, and they practiced with a few words until they felt comfortable for using the proposed system. Later, they were provided with the five phrases mentioned earlier for the experiment. Text phrases were displayed individually, in five runs, on the interface above the text field. Users were instructed to memorize the phrase before a trial started and correctly enter every word, which means the user must correct any misspelled word using the backspace key and cursor navigation keys (if required) in the runtime. In this experiment, the front-end was powered by imaginary left-hand/right-hand motor movements as considered BCI inputs.

5 Results

Following the experimental protocol described in Section 4, a user experiment was conducted, where a total of 10 participants performed text entry tasks for 5 given phrases. The outcomes of the same are analyzed and presented in this section. To quantify the performance of a text entry system, CPM, word per minute, keystroke per character, or the information transfer rate matrices are widely used in literature [4, 26]. In our experiment, the text entry rate (CPM) as CPM_{EF}, and CPM with typing errors (CPM_{WE}) were measured. In addition, EA and CE were measured.

Utilizing the above-defined performance matrices, the average performance of each subject for five phrases is computed and presented in Table 13. In Table 13, the column TT was the addition of IT and CT. The IT represents the average time taken by a user to type a phrase (excluding the time taken for error correction) and the CT represents the average time taken by a user for error correction. The performance matrices EA, CE, CPM_{EF}, and CPM_{WE} were calculated using Equations (2)–(5), respectively.

From the user experiment results, shown in Table 13, the following conclusions are drawn:

(1) The proposed model attains a significantly reasonable text entry rate for all the subjects, with a mean CPM_{WE} of 6.04 and CPM_{EF} of 5.20.
(2) The accuracy in text entry and CE of the proposed model are improved consistently for all the subjects, with a mean EA of 98.04% and mean CE of 0.69, respectively.
(3) The low SD of performance matrices indicates that the current model offers an efficient text entry paradigm across all the participants.

Table 13. Text Entry Performance (Mean over Five Phrases) of Each Subjects

Subjects	TT (Seconds)	EA (%)	CE	CPM_{WE}	CPM_{EF}
Subject-A	340.00	99.05	0.68	6.02	5.59
Subject-B	260.20	99.31	0.75	7.47	6.14
Subject-C	394.65	96.65	0.71	5.47	4.62
Subject-D	505.00	95.89	0.35	3.60	3.03
Subject-E	360.40	98.29	0.69	6.24	5.71
Subject-F	349.80	98.00	0.81	5.83	5.38
Subject-G	241.30	96.49	0.68	7.37	5.92
Subject-H	322.25	97.14	0.75	6.30	5.64
Subject-I	540.15	99.57	0.62	4.96	4.07
Subject-J	285.00	100.00	0.83	7.09	5.88
Mean	359.86 ± 97.88	98.04 ± 1.44	0.69 ± 0.13	6.04 ± 1.18	5.20 ± 0.99

Table 14. Text Entry Performance of Subject-D for Five Phrases

Phrase #	IT (Seconds)	CT (Seconds)	EA (%)	CE	CPM_{WE}	CPM_{EF}
Phrase 1	442	71	96.67	0.50	3.94	3.39
Phrase 2	436	107	100.00	0.27	4.13	3.31
Phrase 3	257	48	93.75	0.40	3.50	2.95
Phrase 4	482	75	92.59	0.25	3.24	2.80
Phrase 5	509	98	96.43	0.33	3.18	2.67
Mean	425.20 ± 98.66	79.80 ± 23.36	95.89 ± 2.88	0.35 ± 0.10	3.60 ± 0.42	3.03 ± 0.32

Table 15. Text Entry Performance of Subject-B for Five Phrases

Phrase #	IT (Seconds)	CT (Seconds)	EA (%)	CE	CPM_{WE}	CPM_{EF}
Phrase 1	248	43	96.55	0.67	6.77	5.77
Phrase 2	226	67	100.00	0.43	8.50	6.55
Phrase 3	98	27	100.00	0.67	9.18	7.20
Phrase 4	251	38	100.00	1.00	6.45	5.59
Phrase 5	261	41	100.00	1.00	6.44	5.56
Mean	216.80 ± 67.63	43.40 ± 14.59	99.31 ± 1.54	0.75 ± 0.25	7.47 ± 1.28	6.14 ± 0.72

(4) The highest performance, error free CPM, attained by the speller is 6.14 char/min (for Subject-B) with 99.31% EA.
(5) The lowest performance, error free CPM, attained by the speller is 3.03 char/min (for Subject-D) with 95.89% EA.

The performance achieved by Subject-B and Subject-D is further analyzed and the individual phrase-wise performance for the lowest performing (Subject-D) and highest performing (Subject-B) participants are shown in Tables 14 and 15, respectively.

In Table 14, the performance of the Subject-D for each phrase is shown. Although EA is high, the CE is low due to a significant amount of time taken in error correction (shown in the column CT).

Table 16. Approximate Percentage of Different Error Correction Policy Used and Percentage of Cursor Navigation Key Used by Each Participants for Phrase-Level/Word-Level Error Correction

	Error Correction Policy Used			Cursor Navigation Key Used
Subjects	PLC (%)	WLC (%)	CLC (%)	for PLC and WLC (%)
Subject-A	9	13	78	92
Subject-B	8	19	73	97
Subject-C	9	27	64	81
Subject-D	11	17	72	57
Subject-E	2	11	87	79
Subject-F	3	14	83	76
Subject-G	9	19	72	93
Subject-H	5	23	72	87
Subject-I	4	17	79	77
Subject-J	2	9	89	94
Mean	6.20 ± 3.36	16.90 ± 5.47	76.90 ± 7.78	83.30 ± 11.99

The higher IT leads to a low entry rate. However, this performance is also considerable and helpful for a motor-impaired person.

Phrase-wise performance of Subject-B is shown in Table 15. From this result it is observed that the proposed model achieved highest performance 9.18 char/min by the Subject-B while entering third phrase. Also, this proposed model attained highest CE as 1.00 and highest accuracy as 100.00% in text entry.

One important design goal of the proposed QC Speller was to reduce users' efforts in error correction and subsequently increase the CE by introducing an efficient error correction policy. Note that the error correction approaches adopted by different users depend on individuals' typing behavior and vary significantly from user to user. In the proposed model, a user can fix an incorrect character using three approaches: **phrase-level correction (PLC)**, **word-level correction (WLC)**, and **character-level correction (CLC)**. Specifically, in PLC, the user corrects one or more misspelled characters after completing the whole phrase. Again, the user fixes misspelled character(s) after completing a wrong word in the WLC approach. In contrast, the user will correct a character immediately after typing it wrong in CLC. An analysis on the user behavior of error correction was performed and reported in Table 16.

Table 16 shows the percentage of error correction approaches followed by each subject and the usage of cursor navigation keys to perform error correction. It was observed that in most cases, the participants corrected errors in character-level and word-level. The last column in this table represents the percentage of cursor navigation keys used by each user in phrase-level and word-level error correction. It is evident from Table 16, the usage of the cursor navigation keys for WLC and PLC (83.30%) is quite favored by most users, leading to higher overall performance. As an example, the subject with the highest overall performance (Subject-B, see Table 13) used the cursor navigation keys for 97% of phrase- and word-level error correction.

6 Discussions

The main objective of this work was to design a BCI speller with an efficient arrangement of symbols/characters. Further, it also would reduce user efforts for error correction. The satisfactory

Table 17. Performance Comparison of Proposed Speller with Virtual Keyboard [28] and Hex-o-Spell [7]

Subject	IT (Seconds)	CT (Seconds)	EA (%)	CE	CPM_{WE}	CPM_{EF}
Virtual Keyboard [28]						
Subject-B	629.60	210.35	72.53	0.27	2.39	2.11
Subject-D	780.50	357.78	58.30	0.19	1.73	1.05
Mean	705.05	284.07	65.42	0.23	2.06	1.58
Hex-o-spell [7]						
Subject-B	464.00	172.50	84.00	0.63	3.67	3.13
Subject-D	593.30	231.15	77.93	0.58	2.35	1.26
Mean	528.65	201.83	80.97	0.61	3.01	2.16
QC Speller (Proposed Method)						
Subject-B	216.80	43.40	99.31	0.75	7.47	6.14
Subject-D	425.20	79.80	95.89	0.35	3.60	3.03
Mean	321.00	61.60	97.60	0.55	5.54	4.59

results obtained in the user interface evaluations and user experiments with complete system fulfilled the above two objectives.

Four prototypes were designed and evaluated; later, Prototype-IV was chosen as the optimal design and integrated with the back-end to build the complete system. A real-time text entry experiment was conducted with 10 participants on the complete system. The proposed paradigm achieved the mean performance of 5.20 char/min with 98.04% accuracy and 6.04 char/min with 2.9% mean error.

To examine the efficacy of the proposed speller comparing with two existing two-class MI-based BCI speller, Virtual Keyboard [28] and Hex-o-spell [7], a comparative analysis was performed in the same experimental paradigm. As the proposed QC Speller and one existing speller [28] are non-predictive speller, hence a non-predictive version of the Hex-o-spell was built in this work. In this experiment, Subject-B and Subject-D participated and performed text entry for the same set of text-phrases using the reconstructed Virtual Keyboard and Hex-o-spell speller. For simplicity and comparing all three BCI spellers under the same paradigm, the same SRC model (described in Section 3.2) was used as the back-end for Virtual Keyboard and Hex-o-spell speller. Results obtained in this experiment are shown in Table 17. From this table, it was observed that proposed speller outperformed the existing two-class MI-based BCI spellers.

Further, the performance of the proposed speller was compared with the reported results of MI signal-based spellers and the results of comparison are shown in the Table 18. Parameters considered in this comparison are the number of target symbols included (#Tar.), number of control commands used (#CC), LM used, ease of error correction, number of keys/operators dedicated for error correction (#CK), maximum entry rate achieved (CPM_{max}), and mean error-free entry rate (CPM_{mean}). From Table 18, it is evident that the proposed speller significantly outperforms the existing two-class MI-based BCI spellers proposed in [7, 28], The primary reason for the high performance of the proposed speller is the efficient symbol arrangement incorporated. Note that, in the mentioned existing spellers letters were arranged in alphabetical order, whereas, in the current study, the symbol arrangement is prioritized by their frequency of use. The high text entry rate achieved substantiates that the approach of an efficient symbol arrangement using the ternary

Table 18. Comparison of the Proposed Speller Performance with the Reported Results of Existing MI-Based BCI Spellers

Reference	Year	#Tar.	#CC	LM	EEC	#CK	CPM_{max}	CPM_{mean}
[28]	2003	32	2	No	Hard	2	1.02	0.85
[36]	2004	28	3	No	Hard	1	3.38	1.99
[7]	2006	30	2	Yes	Hard	2	7.6	4.875
[11]	2012	46	4	Yes	Average	3	3	2.567
[30]	2014	27	3	Yes	Hard	2	3.64	1.68
[8]	2017	47	3	No	Hard	2	NA	9.66
[8]	2017	47	3	Yes	Hard	2	NA	10.15
[44]	2018	27	5	No	Hard	1	6.67	NA
QC Speller	2022	36	2	No	Easy	6	9.18	5.20

Huffman coding algorithm undoubtedly improves the performance of any BCI speller. Again, Table 18 shows that the performance of the proposed speller is comparable with three or more class MI-based spellers proposed in [8, 11, 30, 36, 44].

Another important reason behind the success of the proposed speller is the introduction of an appropriate error correction scheme. By mapping Tables 13 and 16, a positive correlation (Pearson correlation coefficient, $R = 0.696$) between CPM_{EF} and *percentage of cursor navigation key used* is found. Obtained high correlation indicates that higher use of cursor navigation keys for error correction would increase the overall performance. As error-free typing is an important aspect of different forms of formal communication, the error correction policy incorporated in this study can be considered for future BCI speller designs.

The investigation of inter-subject variability of EEG-MI-based spellers has real-world implications for generalized BCI applications [35]. The smaller values for SD for accuracy, CE, and CPM imply that the subject-wise variation of the speller performance is low. The highest performance of 7.47 CPM_{WE} and 6.14 CPM_{EF} was achieved by the model in the case of Subject-B (Table 13), whereas the lowest performance of 3.60 CPM_{WE} and 3.03 CPM_{EF} was obtained in the case of Subject-D. A further investigation of this low performance (Subject-D) is presented in Table 14. It was observed that the inferior performance was mainly due to the high number of errors performed while entering the phrases. Also, from Table 16 it is apparent that the subject rarely used cursor navigation keys for phrase-level and word-level error correction. Integrating a word prediction LM [8] can further improve the performance by reducing the number of typing errors/spelling mistakes.

In Table 13, the mean accuracy value implies on average 98.04% of the presented text was transcribed correctly, and the mean CE implies on average 0.69 incorrect characters were corrected by per correction key (i.e., "backspace," "back," or "cursor navigation keys"). Based on these two performance matrices, we may infer that the proposed speller interface provides an efficient and easy way for error correction. In Section 3.1.4, we discuss how six cursor navigation keys could reduce user efforts for phrase-level and word-level error correction, but these keys would not be helpful in character-level error correction.

7 Conclusions

This work proposed a novel BCI speller interface with a user-friendly error correction method for word-/phrase-level error correction and an efficient symbol arrangement for enhanced speller performance. Along with the backspace and the back keys, six cursor navigation keys introduced in this speller interface reduce users' efforts in error correction and increase CE. This speller interface,

powered by only two mental states, achieves a higher text EA, CE, and text entry rate. The result shows that the proposed speller outperforms existing MI-based BCI spellers. In summary, the proposed system has the potential to support hands-free, touch-free interaction for people with motor impairment for their text entry activities. Again, there is a scope for further improvements in the BCI sub-system by using a hybrid BCI paradigm to identify user intention more precisely. Further, a higher precision-based EEG acquisition device can be used to record high-quality data, leading to an overall improvement. Improvement in the BCI sub-system will increase the overall performance of the text entry.

References

[1] Daniel Afergan, Evan M. Peck, Erin T. Solovey, Andrew Jenkins, Samuel W. Hincks, Eli T. Brown, Remco Chang, and Robert J. K. Jacob. 2014. Dynamic difficulty using brain metrics of workload. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, 3797–3806. DOI: https://doi.org/10.1145/2556288.2557230

[2] Maryam Alimardani and Kazuo Hiraki. 2020. Passive brain-computer interfaces for enhanced human-robot interaction. *Frontiers in Robotics and AI* 7 (2020), 125. DOI: https://doi.org/10.3389/frobt.2020.00125

[3] Sandra Alper and Sahoby Raharinirina. 2006. Assistive technology for individuals with disabilities: A review and synthesis of the literature. *Journal of Special Education Technology* 21, 2 (2006), 47–64. DOI: https://doi.org/10.1177/016264340602100204

[4] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2009. Analysis of text entry performance metrics. *IEEE Toronto International Conference - Science and Technology for Humanity* (2009), 100–105. DOI: https://doi.org/10.1109/TIC-STH.2009.5444533

[5] Annushree Bablani, Damodar Reddy Edla, Diwakar Tripathi, and Ramalingaswamy Cheruku. 2019. Survey on brain-computer interface: An emerging computational intelligence paradigm. *ACM Computing Surveys* 52, 1, Article 20 (2019), 32 pages. DOI: https://doi.org/10.1145/3297713

[6] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kübler, J. Perelmouter, E. Taub, and H. Flor. 1999. A spelling device for the paralysed. *Nature* 398, 6725 (1999), 297–298. DOI: https://doi.org/10.1038/18581

[7] B. Blankertz, G. Dornhege, Matthias Krauledat, Michael Tangermann, J. Williamson, Roderick Murray-Smith, and Klaus-Robert Müller. 2006. The Berlin brain-computer interface presents the novel mental typewriter Hex-o-spell. *Clinical Neurophysiology* 113 (01 2006). Retrieved from https://api.semanticscholar.org/CorpusID:2653216

[8] Lei Cao, Bin Xia, Oladazimi Maysam, Jie Li, Hong Xie, and Niels Birbaumer. 2017. A synchronous motor imagery based neural physiological paradigm for brain computer interface speller. *Frontiers in Human Neuroscience* 11 (2017). DOI: https://doi.org/10.3389/fnhum.2017.00274

[9] John P. Chin, Virginia A. Diehl, and Kent L. Norman. 1988. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '88)*. ACM, 213–218. DOI: https://doi.org/10.1145/57167.57203

[10] Rachel E. Cowan, Benjamin J. Fregly, Michael L. Boninger, Leighton Chan, Mary M. Rodgers, and David J. Reinkensmeyer. 2012. Recent trends in assistive technology for mobility. *Journal of NeuroEngineering and Rehabilitation* 9, 1 (2012), 20. DOI: https://doi.org/10.1186/1743-0003-9-20

[11] Tiziano D'Albis, Rossella Blatt, Roberto Tedesco, and Licia Sbattella. 2012. A predictive speller controlled by a brain-computer interface based on motor imagery. *ACM Transactions on Computer-Human Interaction* 19, 3 (2012), 20–25. DOI: https://doi.org/10.1145/2362364.2362368

[12] Canadian Centre for Occupational Health Government of Canada and Safety. 2023. Office Ergonomics - Positioning the Monitor. Retrieved from https://www.ccohs.ca/oshanswers/ergonomics/office/monitor_positioning.html

[13] Ben D. Harper and Kent L. Norman. 1993. Improving user satisfaction: The questionnaire for user interaction satisfaction version 5.5. In *Proceedings of the 1st Annual Mid-Atlantic Human Factors Conference*, 224–228. Retrieved from https://api.semanticscholar.org/CorpusID:60950368

[14] Bin He, Bryan Baxter, Bradley J. Edelman, Christopher C. Cline, and Wenjing W. Ye. 2015. Noninvasive brain-computer interfaces based on sensorimotor rhythms. *Proceedings of the IEEE* 103, 6 (2015), 907–925. DOI: https://doi.org/10.1109/jproc.2015.2407272

[15] W. E. Hick. 1952. On the rate of gain of information. *Quarterly Journal of Experimental Psychology* 4, 1 (1952), 11–26. DOI: https://doi.org/10.1080/17470215208416600

[16] David A. Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE* 40, 9 (1952), 1098–1101. DOI: https://doi.org/10.1109/JRPROC.1952.273898

[17] R. Hyman. 1953. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology* 45, 3 (1953), 188–196. DOI: https://doi.org/10.1037/H0056940

[18] Blake Ives, Margrethe H. Olson, and Jack J. Baroudi. 1983. The measurement of user information satisfaction. *Communications of the ACM* 26, 10 (1983), 785–793. DOI: https://doi.org/10.1145/358413.358430

[19] Zhenrui Ji, Quan Liu, Wenjun Xu, Bitao Yao, Jiayi Liu, and Zude Zhou. 2021. A closed-loop brain-computer interface with augmented reality feedback for industrial human-robot collaboration. *The International Journal of Advanced Manufacturing Technology* (2021). DOI: https://doi.org/10.21203/rs.3.rs-283263/v1

[20] Nataliya Kosmyna, Franck Tarpin-Bernard, and Bertrand Rivet. 2015. Conceptual priming for in-game BCI training. *ACM Transactions on Computer-Human Interaction* 22, 5, Article 26 (2015), 25 pages. DOI: https://doi.org/10.1145/2808228

[21] G. Kouroupetroglou. 2013. *Assistive Technologies and Computer Access for Motor Disabilities*. Medical Information Science Reference, 433 pages. Retrieved from https://api.semanticscholar.org/CorpusID:107913525

[22] Jyotish Kumar and Jyoti kumar. 2016. Affective modelling of users in HCI using EEG. *Procedia Computer Science* 84 (2016), 107–114. DOI: https://doi.org/10.1016/j.procs.2016.04.073

[23] Shiu Kumar, Alok Sharma, and Tatsuhiko Tsunoda. 2017. An improved discriminative filter bank selection approach for motor imagery EEG signal classification using mutual information. *BMC Bioinformatics* 18, 16 (2017), 545. DOI: https://doi.org/10.1186/s12859-017-1964-6

[24] Yifeng Li and Alioune Ngom. 2013. Sparse representation approaches for the classification of high-dimensional biological data. *BMC Systems Biology* 7, 4 (2013). DOI: https://doi.org/10.1186/1752-0509-7-S4-S6

[25] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *Proceedings of the Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, 754–755. DOI: https://doi.org/10.1145/765891.765971

[26] I. Scott MacKenzie and Kumiko Tanaka-Ishii. 2007. *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufmann Publishers Inc.

[27] OpenBCI Newsletter. 2020. *Spir-o-Spell, an Optimized MI-Based BCI Speller*. Technical Report.

[28] B. Obermaier, G. R. Müller, and G. Pfurtscheller. 2003. "Virtual keyboard" controlled by spontaneous EEG activity. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 11, 4 (12 2003), 422–426. DOI: https://doi.org/10.1109/TNSRE.2003.816866

[29] Jonathan Peirce, Jeremy R. Gray, Sol Simpson, Michael MacAskill, Richard Höchenberger, Hiroyuki Sogo, Erik Kastman, and Jonas Kristoffer Lindeløv. 2019. PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods* 51, 1 (2019), 195–203. DOI: https://doi.org/10.3758/s13428-018-01193-y

[30] S. Perdikis, R. Leeb, J. Williamson, A. Ramsay, M. Tavella, L. Desideri, E.-J. Hoogerwerf, A. Al-Khodairy, R. Murray-Smith, and J. D. R. Millan. 2014. Clinical evaluation of BrainTree, a motor imagery hybrid BCI speller. *Journal of Neural Engineering* 11, 3 (2014), 036003. DOI: https://doi.org/10.1088/1741-2560/11/3/036003

[31] Kevin M. Pitt, Jonathan S. Brumberg, Jeremy D. Burnison, Jyutika Mehta, and Juhi Kidwai. 2019. Behind the scenes of noninvasive brain computer interfaces: A review of electroencephalography signals, how they are recorded, and why they matter. *Perspectives of the ASHA Special Interest Groups* 4, 6 (2019), 1622–1636. DOI: https://doi.org/10.1044/2019_PERS-19-00059

[32] Danny Plass-Oude Bos, Boris Reuderink, Bram Laar, Hayrettin Gürkök, Christian Mühl, Mannes Poel, Dirk Heylen, and Anton Nijholt. 2010. Human-computer interaction for BCI games usability and user experience. In *Proceedings of the 2010 International Conference on Cyberworlds (CW '10)*, 277–281. DOI: https://doi.org/10.1109/CW.2010.22

[33] Rajesh P. N. Rao. 2013. *Brain-Computer Interfacing: An Introduction*. Cambridge University Press.

[34] Aya Rezeika, Mihaly Benda, Piotr Stawicki, Felix Gembler, Abdul Saboor, and Ivan Volosyak. 2018. Brain–computer interface spellers: A review. *Brain Sciences* 8, 4 (2018). DOI: https://doi.org/10.3390/brainsci8040057

[35] Simanto Saha and Mathias Baumert. 2020. Intra- and inter-subject variability in EEG-based sensorimotor brain computer interface: A review. *Frontiers in Computational Neuroscience* 13 (2020). DOI: https://doi.org/10.3389/fncom.2019.00087

[36] R. Scherer, G. R. Muller, C. Neuper, B. Graimann, and G. Pfurtscheller. 2004. An asynchronously controlled EEG-based virtual keyboard: Improvement of the spelling rate. *IEEE Transactions on Biomedical Engineering* 51, 6 (2004), 979–984. DOI: https://doi.org/10.1109/TBME.2004.827062

[37] S. R. Sreeja, J. Rabha, D. Samanta, P. Mitra, and M. Sarma. 2017. Classification of motor imagery based EEG signals using sparsity approach. In *Intelligent Human Computer Interaction*. Patrick Horain, Catherine Achard, and Malik Mallem (Eds.), Springer International Publishing, 47–59. DOI: https://doi.org/10.1007/978-3-319-72038-8_5

[38] S. R. Sreeja and D. Samanta. 2019. Classification of multiclass motor imagery EEG signal using sparsity approach. *Neurocomputing* 368 (2019), 133–145. DOI: https://doi.org/10.1016/j.neucom.2019.08.037

[39] Bram van de Laar, Hayrettin Gürkök, Danny Plass-Oude Bos, Mannes Poel, and Anton Nijholt. 2013. Experiencing BCI control in a popular computer game. *IEEE Transactions on Computational Intelligence and AI in Games* 5, 2 (2013), 176–184. DOI: https://doi.org/10.1109/TCIAIG.2013.2253778

[40] K. Venu and P. Natesan. 2021. Review on motor imagery based EEG signal classification for BCI using deep learning techniques. In *Advances in Intelligent Systems and Computing*. Jessnor Arif Mat Jiza, Ismail Mohd Khairuddin, Mohd Azraai Mohd Razman, Ahmad Fakhri Ab. Nasir, Mohamad Shaiful Abdul Karim, Abdul Aziz Jaafar, Lim Wei Hong, Anwar P. P. Abdul Majeed, Pengcheng Liug, et al. (Eds.), Springer International Publishing, 1350, 137–154. DOI: https://doi.org/10.1007/978-3-030-70917-4_15

[41] John Williamson and Roderick Murray-Smith. 2005. Hex: Dynamics and probabilistic text entry. In *Switching and Learning in Feedback Systems*. R. Murray-Smith and R. Shorten (Eds.), Lecture Notes in Computer Science, Vol. 3355, Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-540-30560-6_15

[42] Bin Xia, Jing Yang, Conghui Cheng, and Hong Xie. 2013. A motor imagery based brain-computer interface speller. In *Advances in Computational Intelligence*. Ignacio Rojas, Gonzalo Joya, and Joan Cabestany (Eds.), Springer, Berlin, 413–421.

[43] Drishti Yadav, Shilpee Yadav, and Karan Veer. 2020. A comprehensive assessment of brain computer interfaces: Recent trends and challenges. *Journal of Neuroscience Methods* 346 (2020), 108918. DOI: https://doi.org/10.1016/j.jneumeth.2020.108918

[44] Xiang Zhang, Lina Yao, Quan Z. Sheng, Salil S. Kanhere, Tao Gu, and Dalin Zhang. 2018. Converting your thoughts to texts: Enabling brain typing via deep feature learning of EEG signals. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Athens, Greece, 1–10. DOI: https://doi.org/10.1109/PERCOM.2018.8444575

Received 7 February 2024; revised 3 September 2024; accepted 14 November 2024