

Clinical evaluation of BrainTree, a motor imagery hybrid BCI speller

S Perdikis¹, R Leeb¹, J Williamson², A Ramsay², M Tavella¹, L Desideri³, E-J Hoogerwerf³, A Al-Khodairy⁴, R Murray-Smith² and J d R Millán¹

¹ Chair in Non-Invasive Brain-Machine Interface, Center for Neuroprosthetics, School of Engineering, École Polytechnique Fédérale de Lausanne, Station 11, 1015 Lausanne, Switzerland

² Inference, Dynamics and Interaction Group, School of Computing Science, University of Glasgow, Glasgow, United Kingdom

³ Ausilioteca, AIAS Bologna onlus, Bologna, Italy

⁴ Clinique Romande de Réadaptation Suvacare, Sion, Switzerland

E-mail: robert.leeb@epfl.ch, jose.millan@epfl.ch

Abstract. *Objective.* While brain-computer interfaces (BCIs) for communication have reached considerable technical maturity, there is still a great need for state-of-the-art evaluation by end-users outside laboratory environments. To achieve this primary objective, it is necessary to augment a BCI with a series of components that allow end-users to type text effectively. *Approach.* This work presents the clinical evaluation of a motor imagery (MI) BCI text-speller, called BrainTree, by 6 severely disabled end-users and 10 able-bodied users. Additionally, we define a generic model of code-based BCI applications which serves as an analytical tool for evaluation and design. *Main results.* We show that all users achieved remarkable usability and efficiency outcomes in spelling. Furthermore, our model-based analysis highlights the added value of human-computer interaction (HCI) techniques and hybrid BCI error-handling mechanisms, and reveals the effects of BCI performances on usability and efficiency in code-based applications. *Significance.* This study demonstrates the usability potential of code-based MI spellers, with BrainTree being the first to be evaluated by a substantial number of end-users, establishing them as a viable, competitive alternative to other popular BCI spellers. Another major outcome of our model-based analysis is the derivation of a 80% minimum command accuracy requirement for successful code-based application control, revising upwards previous estimates attempted in the literature.

Keywords: brain-computer interface, clinical evaluation, text-speller, motor imagery, human computer interaction, hybrid BCI

To appear in: *J. Neural Eng.*

1. Introduction

Brain-computer interface (BCI) technology has been established as a promising solution for enabling communication capabilities to people with motor function impairments [1]. Among all demonstrated BCI applications, text-entry systems hold a special place in view of the fact that improving their communication channels is the top priority of severely disabled people. A first landmark in BCI dealt with a spelling device operated by amyotrophic lateral sclerosis (ALS) patients by means of learned slow cortical potential (SCP) modulation [2]. The possibility to exploit other brain phenomena was soon also demonstrated, i.e. event-related potentials (ERPs) [3, 4], sensorimotor rhythms (SMR) [5, 6], or low-frequency asynchronous switch design (LF-ASD) [7].

Recent efforts have focused on the optimization of the above BCI paradigms for text spelling [8]. ERP-based text-entries and especially the P300 paradigm have prevailed (reviews in [9] and [10]), however, other types of BCI spellers have also attracted attention and undergone similar optimization procedures, such as motor imagery (MI) [11, 12], steady-state visually evoked potentials (SSVEP) [13], visual attention [14] and rapid serial visual presentation (RSVP) [15] systems.

Despite the vast amount of literature dedicated to BCI-actuated spellers, the main limitation of the state-of-the-art remains the lack of clinical evaluation of BCI spelling prototypes by end-users. To the best of our knowledge, out of dozens related studies, only fifteen report results on experiments involving disabled individuals. Nine of these studies are based on P300: 16 end-users in [16], 11 in [17], 10 in [18], 6 in [19], 5 in [20], 4 in [21], 3 in [4] and [22], 1 in [23]. Two are based on SCPs: 5 in [2] and 2 in [24]. One study for other BCI modalities: LF-ASD with 3 end-users [7], 1 in MI [6] and 1 in RSVP [25]. P300-spelling case studies with a single, non-disabled individual have been also reported with invasive BCIs [26, 27]. This situation poses a major obstacle into assessing the actual level and limitations of BCI spelling for the intended population and hinders the crucial step of transferring this technology out of the lab, especially for paradigms besides P300.

The main contribution of the present article is reporting on the clinical evaluation of a novel MI-based hybrid BCI spelling prototype, called *BrainTree*, by 6 disabled end-users and 10 able-bodied individuals. To our knowledge, this is the first clinical evaluation of a motor imagery BCI speller by a fairly large pool of end-users. The motivation behind the selection of the MI BCI paradigm is twofold: First, it is well known that not all subjects can successfully operate any given BCI paradigm (including P300 spellers, [28, 29] – it is therefore necessary to provide a wider variety of BCI solutions to suit the specific user abilities and preferences. The MI BCI paradigm has been successfully employed in numerous applications, thus MI spellers are a natural candidate for providing such an alternative solution. Second, the current relative superiority of P300 in terms of typing speed is compromised by its (conventionally) synchronous and dependent on external stimuli operation modality [1],

while MI BCIs offer the possibility of self-paced and independent control during spelling which many users assess as more natural [30, 7]. While a few recent works demonstrate the possibility of asynchronous P300-based operation [31, 32, 33], MI remains the natural option for self-paced interaction, as discussed in [34, 35].

In our study, we have initially attempted to integrate our 2-class MI BCI into a commercial assistive technology (AT) software based on a standard timing-based paradigm (scanning mode). Thereby, each target is encoded through the timing at which BCI commands are delivered relatively to other interface events and the spatial configuration of targets (like in [11, 12]). Timing-based features are introduced to optimize efficiency, however, such paradigms require additional BCI skills like fast transitions between mental states, extensive ability to sustain those, or, the ability to stay idle (intentional non-control state, INC) for long intervals. Thus they might have a negative impact on the application’s usability, in an implicit trade-off. Although a preliminary test with 3 able-bodied subjects was successful [36], evaluation by 3 end-users exhibited a compromised usability of this initial prototype. Following a user-centered approach, this quantitative evaluation together with qualitative feedback from end-users led to the introduction of the *code-based* BrainTree speller, where each target is represented by a codeword in the dictionary of defined mental states eliminating timing-based features. Importantly, we show that all 16 participants achieved remarkable usability and competitive efficiency outcomes with our BrainTree speller.

Finally, a key ingredient for the successful operation of the BrainTree speller is that we have enhanced it with data-compression human-computer interaction (HCI) mechanisms exploiting contextual information provided through a language model, as well as an error-handling mechanism following a hybrid BCI approach [37, 38], based in this case on electromyographic (EMG) signal monitoring. To formalise the contributions of these two features and theoretically establish the importance and derived minimum requirements of the basic BCI performances, we have defined a model for the analysis of code-based BCI applications, which is generic enough to be applicable in any code-based application for design optimization.

2. Methods

This section describes the participants and the different stages of our user evaluation study, as well as the biosignal acquisition, experimental environment and apparatus. It also details the BrainTree spellers interface, control paradigm, and embedded HCI principles. Subsequently, we summarize the methods employed for the implementations of the MI BCI and EMG monitoring. Finally, we present the experimental protocol used in the spelling sessions.

2.1. Participants and training protocol

The BrainTree user evaluation study comprises two basic phases, the BCI training sessions and the spelling sessions. The two phases were separated by variable time periods for each subject, ranging from one week to several months. The training phase begins with one or more “offline” calibration sessions to collect data for building a subject-specific MI BCI classifier. More specifically, the best pair out of 3 exercised MI tasks (right, left hand and feet) is selected and the corresponding data used to train the classifier. The selection of the best two MI classes limits the spelling application paradigm to binary spellers. Yet, it ensures the maximization of the overall BCI accuracy, as it is well known that for the vast majority of users the latter is severely degraded by attempting to discriminate among additional brain patterns. All subjects were left free to choose the kinesthetic mental strategies they felt more comfortable with in a short rehearsal before training, although repetitive “ball squeezing” hand movements or “tennis playing” swinging arm movements and “kicking” feet movements were provided as examples [39]. Out of 16 participants (see below) of the BrainTree spelling study 6 employed some variation of right and left hand MI (2 end-users and 4 able-bodied-users), 5 right hand and feet MI (2 end-users and 3 able-bodied users), and another 5 used left hand and feet MI (2 end-users and 3 able-bodied users). Optimal MI pairs are thus almost uniformly distributed among users and the two user categories.

Subsequently, subjects proceed with “online”, closed-loop MI BCI sessions where they adapt to the BCI by learning to voluntarily modulate the EEG oscillatory rhythms. The overall training procedure involved at most 9 BCI sessions executed once or twice per week, where a single session lasted at most 2 hours including preparation and setup. Further details on the training phase can be found in [39].

BrainTree spelling sessions were carried out according to the spelling protocol described in Section 2.6. All users completed the spelling protocol within one or two sessions, whereby each session lasted 1-2 hours including preparation, setup and intermissions. An EMG calibration protocol (~ 2 minutes) took place right before a spelling session in order to estimate the EMG detection threshold th_{EMG} , Section 2.5.

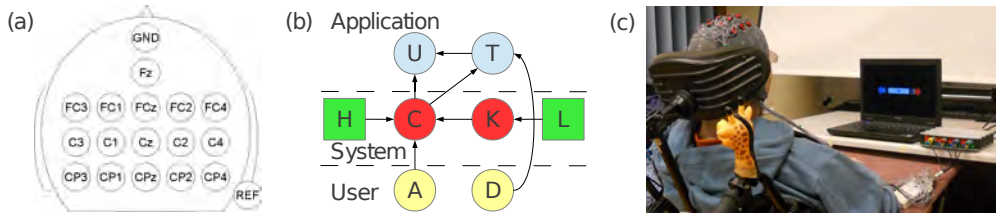


Figure 1: (a) EEG configuration, (b) Code-based application model, (c) End-user training.

A total of 6 end-users (E3–E8, Table 1), and 10 able-bodied users (H1–H10, 9 male and 1 female, 29.0 ± 4.6 of age) participated in the final BrainTree evaluation. Another

2 end-users (E1–E2) and 4 able-bodied users (H11–H14, 28.5 ± 1.9 of age) participated in the initial timing-based prototype evaluation (see Section 1), while end-user E3 tried both prototypes. The study was approved by the local Ethics Committee and participants signed the informed consent forms. None of the end-user participants suffered brain trauma or cognitive impairments.

Table 1: Details of end-user participants

ID	Sex	Age	Diagnosis	MI tasks
E1	F	34	Myopathy	right hand, feet
E2	M	30	Spinal amyotrophy 2	left hand, feet
E3	M	60	C6 tetraplegia	left hand, feet
E4	M	43	C5–C6 tetraplegia	left hand, feet
E5	M	41	C6 complete tetraplegia	right and left hand
E6	M	23	C7 tetraplegia	right hand, feet
E7	M	48	C6 tetraplegia	right hand, feet
E8	M	19	Duchenne dystrophy	right and left hand

It should be noted that 24 end-users underwent the prerequisite MI training phase [39] as part of a larger BCI prototype user evaluation study. However, approximately 50% of them qualified for the study’s follow-up, because they reached the required inclusion criterion of 70% command accuracy within the limited training period of up to 9 BCI sessions, as defined in the study’s protocol. The rest of the initial participants either quit the training phase for personal or technical reasons, or successfully completed it but did not proceed with BrainTree spelling sessions due to unavailability at the scheduled time or for logistic reasons (2 of them only participated in the initial spelling prototype evaluation as mentioned above). None of the 6 end-users that participated in BrainTree evaluation had undergone more than 5 BCI training sessions prior to the spelling sessions.

2.2. Biosignal data acquisition and experimental apparatus

Biosignals are recorded using a g.USBamp amplifier (g.tec medical engineering, Schiedelberg, Austria) at 512 Hz sampling rate, band-pass filtered between 0.1 Hz and 100 Hz and notch filtered at 50 Hz. The brain activity is acquired via 16 active EEG channels (g.GAMMAbox) over the sensorimotor cortex (exact channel locations in Figure 1a). A second g.USBamp system is additionally employed during spelling sessions to synchronously record a bipolar EMG channel.

The muscle *extensor carpi radialis longus* on the user’s right or left forearm, depending on which limb is not employed for MI, has been targeted for most end-users that maintained

at least limited mobility of one of the forearms. For certain able-bodied users, as well as for end-user E5, the same limb has been used for both MI and EMG monitoring, however any interference is irrelevant since the primary and secondary control modalities are never employed at the same time. For end-user E4 the muscle *trapezius* has been employed. Besides that, in the case of user E8 a button was used for “undo” commands instead of the EMG-based functionality, due to spasticity that generated false positive EMG detections, which however left the button functionality unaffected.

A single laptop is employed for both training and spelling sessions, displaying the corresponding training or spelling interface. Able-bodied subjects were comfortably seated approximately 1m away from the monitor. End-users were seated in their wheelchair at approximately the same distance from the monitor, see Figure 1c. End-user E7 was bedridden at the time of the evaluation and a bed table has been employed. All experiments were performed in real-world conditions involving noisy and crowded environments in clinics, in the lab or at the end-user’s home. Most importantly, no classifier re-training or adaptation took place before spelling. The classifiers were typically trained one week to several months before the spelling session(s), thus our results reflect the ability of subjects to operate our speller “on-demand” to flawlessly type messages.

2.3. Graphical User Interface, underlying structure and control paradigm of the text-entry system

BrainTree is a binary speller allowing text input using only two commands. Efficient character selection is achieved thanks to an underlying binary tree structure hidden behind a simple Graphical User Interface (GUI, Figure 2a). Given a typed prefix (including the empty one in the beginning), the speller generates a new *binary tree*, in which characters/targets are leaf nodes. A binary codeword is thus associated to each character, where a left branch towards the character’s node becomes 0 and each right branch 1. The user descends the tree with left/right transitions through a 2-class MI BCI.

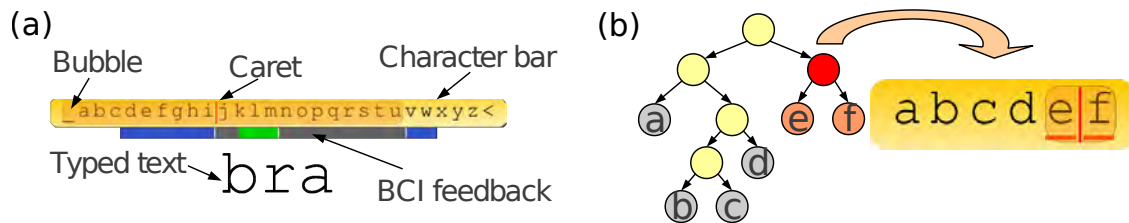


Figure 2: (a) GUI and (b) underlying binary tree structure in a reduced dictionary example.

Figure 2b provides an example of the association between the GUI and the underlying binary tree structure on a reduced dictionary. The current node is visualized by the

red cursor (caret) and the available characters are surrounded by the orange “bubble”. The user only has to identify the position of the desired character relatively to the caret, simplified by alphabetic ordering and thus inflicting minimal mental workload, and forward the appropriate BCI command to move the caret towards it.

An “undo” functionality complements the primary control modality (tree descent through the BCI), allowing the user to return to the previous caret position (ascend to the parent node) after an erroneous BCI command. This third command, useful upon error commitment and thus only infrequently employed, is implemented by means of Electromyographic (EMG) signal monitoring and is activated by brisk movements of one of the user’s muscles (normally limbs). It represents a hybrid BCI approach which exploits potential residual muscle activity of a disabled user. However, this choice, which has been based on our end-user pathological profiles, is of no particular significance and can be replaced by any other “switch” an end-user might be able to operate, as discussed in more detail in Section 5. The availability of a “backspace” character provides a second error-handling approach by deleting the last typed character. The profound difference between “undo” and “backspace” is that the former allows immediate deletion of only the last digit/command of the desired character’s codeword through tree ascent, while the latter deletes the whole last typed codeword/character.

Finally, a conventional MI feedback bar is embedded into the BrainTree GUI (Figure 2a) to continuously visualize the user’s intent. A command is forwarded when the bar reaches the left/right threshold.

2.4. Context-aware data compression

BCI spellers are faced with the challenge of mapping a limited number of recognized mental states into a much larger number of targets (e.g. latin characters). In order to increase spelling efficiency, code-based paradigms (having eliminated timing-based features) can only rely on data compression HCI techniques. This is formally achieved with entropy coding algorithms that reduce the length of target codewords. However, such algorithms require contextual information in the form of a probability distribution over the target set. In BrainTree, the latter is provided through a prefix-based language model (smoothed prediction by partial matching, PPM). The model is learned from a set of source corpora (an English e-book has been used here). The prefix-dependent nature of PPM results in a new character distribution and, hence, a new binary tree for each character selection round. While the Huffman algorithm guarantees the minimum average code length, for our study the Hu-Tucker algorithm [40] is preferred, since it is nearly optimal (no code will be more than 1 bit longer than the equivalent Huffman), while respecting a (here, alphabetic) lexicographic ordering. The advantage is that characters are not shuffled after a letter insertion, which would greatly confuse the user.

2.5. MI BCI and EMG processing

The main control modality for left/right commands in BrainTree is implemented by means of a 2-class MI BCI, detailed in [39]. EEG signals are spatially filtered with a local Laplacian derivation and the power spectral density (PSD) is estimated (4–48 Hz with 2 Hz resolution) over the last second. The PSD is computed every 62.5 ms (i.e., 16 times per second) using the Welch method (five 25%–overlapping internal Hanning windows of 500 ms) and log-transformed. The overall feature vectors are thus spectral density estimates on combinations of channels and frequency bands. A small subset of features (usually 5–10) is selected as described in [39]. Classification of the reduced PSD feature vectors is achieved using a Gaussian Mixture Model (GMM) framework which outputs a posterior probability distribution on the mental classes. The classifier probabilities are integrated with an exponential smoothing filter, the output of which is visualized through BrainTree’s BCI feedback bar (see Figure 2a); a left/right command is finally delivered by thresholding the smoothed integration output.

The secondary control modality for “undo” commands is implemented through monitoring a single bipolar EMG channel for the detection of a user’s brisk movement. EMG data are acquired and processed synchronously to the EEG at an output rate of 16 Hz. The EMG signal is bandpass filtered (30–40 Hz, Butterworth) and the signal envelope is computed by means of the absolute value of the signal’s Hilbert transform. The envelope is averaged within a 1 sec-long window to provide a scalar EMG feature. EMG features are thresholded through a detection threshold th_{EMG} to provide an initial detection output (0/1). Threshold th_{EMG} is calculated during the short EMG calibration phase by collecting ten, 1 second-long examples of the user’s selected movement, as: $th_{EMG} = MaxOff + 0.5 \times (AvgOn - MaxOff)$, where $MaxOff$ the maximum observed EMG feature value during rest and $AvgOn$ the average feature value during movement execution. The original detection outputs are afterwards exponentially smoothed to avoid false positives by potential slight, unintentional user’s movements. An “undo” command is finally delivered by thresholding the integrated output with a fixed decision threshold $th_d = 0.6$.

2.6. Experimental spelling protocol

The BrainTree evaluation protocol consists of four copy-spelling tasks (“hello”, “email”, “computer” and “internet”, executed in this order), repeated across three different conditions. The tasks have been chosen to represent common words in the English dictionary, so as to comply with the PPM language model of BrainTree. A task is only considered successful when the whole word is written without errors. No time constraints are imposed for task completion. The first condition, referred to as $hBCI+CA$, is meant to evaluate the overall usability and efficiency of BrainTree, where both language model, context-aware data compression and hybrid-based “undo” error-handling are available. The $hBCI$ (context-

awareness disabled using prefix-independent, uniform probability distribution of characters) and *CA* (“undo” command disabled) conditions are introduced to experimentally quantify the added value of data compression HCI and “undo” error-handling by disabling the respective features. The three conditions are executed in the order mentioned here. Due to time limitations in clinics, only two out of six end-users performed the *CA* condition.

3. General model of code-based BCI applications

We theoretically establish the usability and efficiency of code-based applications introducing a general model, which serves as a tool to study the relative added value of data compression techniques and error-handling mechanisms as well as the minimum requirements of BCI control-related metrics, and their effects on the application.

Figure 1b illustrates a graphical representation of the probabilistic model for target selection in code-based applications. Variable $U \in \{0, 1\}$ expresses the event of target selection and T the amount of time needed, thus relating the model to the application’s usability (as target selection ability) and efficiency, respectively. We further assume that the probability of succeeding in typing a character $p(U = 1)$ depends on subjective thresholds t_c on a variable C which reflects the needed number of commands (inserted codeword digits) until selection as well as t_t on T , so that $p(U = 1) = p((C \leq t_c) \cap (T \leq t_t))$. K represents the number of minimum required commands for a selection (codeword length), A the subject’s average command accuracy and D delivery time. Finally, variables H and L reflect the type of error handling mechanisms (H) and existence of context-aware data compression (L), respectively. It should be underlined that accuracy A and delivery time D (reflecting BCI speed) are the only BCI-related metrics involved in a code-based paradigm.

As discussed in [41], error-handling in coding-based applications can either rely on the possibility to “undo” an erroneously inserted digit in the target’s codeword ($H = 1$) or on the availability of a “delete” target ($H = 0$) which deletes the whole last codeword (like BrainTree’s “backspace”). Expectations on the defined variables can be derived exploiting the nature of coding-based schemes and known results from probability theory. The expectation on the number of required commands, $E[C]$, can be approximated as:

$$E[C] = \begin{cases} \frac{K}{A} & , \forall A, H = 1 \\ \frac{1-A^K}{A^K}(G + O) + K & , A \geq \sqrt[M]{0.5}, H = 0 \\ \infty & , A < \sqrt[M]{0.5}, H = 0 \end{cases} \quad (1)$$

assuming a perfect “undo”. G expresses the number of commands needed after an error so that the user is in position to start reaching the “delete” target, while O is the number of commands needed to bring the application at the state exactly before the erroneous codeword insertion and M the known length of the “delete” codeword. The expectations on the time

needed to correctly complete a codeword, $E[T]$, can then be derived as:

$$E[T] = \begin{cases} E_{A,K}[C](D + P) + E_{A,K}[C_{undo}]P & , H = 1 \\ E_{A,K}[C](D + P) & , H = 0 \end{cases} \quad (2)$$

where $E[C_{undo}] = ((1 - A)/A)K$ and P the duration of a “pause” applied in-between consecutive commands ($P=3$ sec in our study). While derivations for $E[T]$ are straightforward, those of Equation 1 are justified and extended in Appendix A.

4. Results

4.1. Experimental results

We evaluate BrainTree’s text entry usability through task completion rates, see Figure 3a. Across all 172 attempted tasks (for all users in all conditions), an impressive 94.2% completion rate is demonstrated (98.2% for end-users and 92.2% for able-bodied users). The most remarkable finding is the 100% task completion success for both user categories when considering the *hBCI+CA* condition. Rates are slightly compromised in *hBCI* with 96.9% (95.8% for end-users and 97.5% for able-bodied users) and more significantly affected in *CA* with 81.8% (100% only for two end-users and 77.8% for nine able-bodied users).

Concerning efficiency, Figure 3b illustrates the typing speeds (as seconds per character, spc). Users were able to type on average at 35.7 spc (1.68 characters per minute, cpm) in *hBCI+CA*, 44.85 spc (1.34 cpm) in *hBCI* and 34.18 spc (1.76 cpm) in *CA*. The fastest tasks were in 16.5 spc (3.64 cpm) in *hBCI+CA* and *CA* and 22.6 spc (2.65 cpm) in *hBCI*. Thus, 1.68 cpm is what can be expected for BrainTree by an average user (1.47 cpm for end-users and 1.84 cpm for able-bodied participants in *hBCI+CA*) and 3.6 cpm a validated typing speed upper bound estimate (achieved by end-user E8 in *CA* and user H8 in *hBCI+CA*).

Typing speed results show high variability (1–2.6 cpm across subjects for *hBCI+CA*) as well as within subjects (high standard deviations in Figure 3b), also observed in [11, 12, 41]. This variability is attributed to the fluctuations of BCI command accuracy A , Figure 3c and delivery time D , Figure 3d, as detailed by the large within-subject value ranges and the substantial across-subject median differences for both A and D . Consequently, the benefits of HCI and error-handling methods are experimentally not obvious in the average typing speeds per condition. We therefore establish a methodological paradigm for studying the effects of BCI performances and contributions of assistive mechanisms in code-based BCI applications exploiting our model definition and applying sensitivity analysis.

4.2. Model validation

We validate our model by comparing in Figure 4a our theoretically derived expectations on the number of commands needed ($E[C]$, Equation 1) with the measured values, against

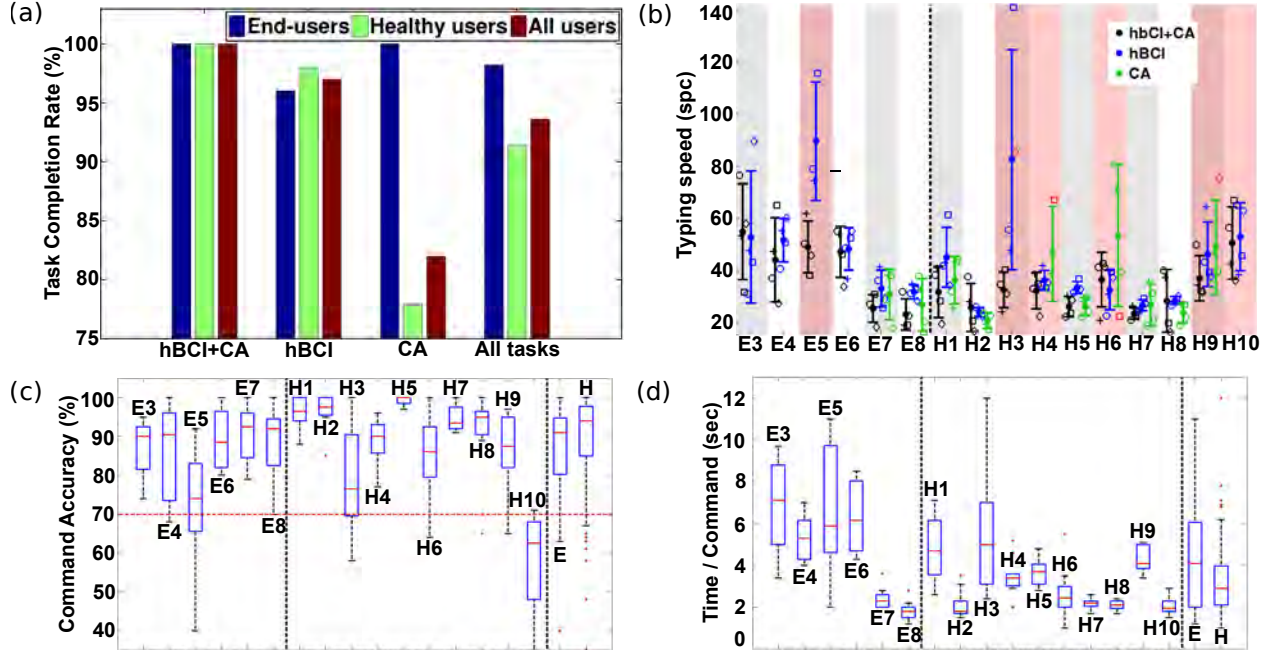


Figure 3: Top: (a) Task completion rates and (b) average typing speed performances (per task), \circ for “hello”, $+$ for “email”, \square for “computer” and \diamond for “internet”. Mean subject speed per condition is shown with solid points and standard deviation with errorbars. Performance in a failed task is shown in red. Red-shaded areas indicate subjects with incomplete or unexecuted tasks. Bottom: Boxplots (median and 25th, 75th percentiles) of average command accuracy (c) and command delivery time (d). Boxes E and H show grand averages computed on all tasks performed by end-users and able-bodied users, respectively.

command accuracy A and substituting $E[K]$ measured to be 3.8 ± 1.0 with and 4.9 ± 0.3 without context-awareness for the words used in our protocol. The measured command number per character for different tasks (failed ones shown in red) match very well our model-based theoretical predictions. The model also helps to explain the experimental usability and typing speed results and make the contributions of data-compression and “undo” error-handling more clear. The “undo” option (black for $hBCI+CA$ and blue for $hBCI$ in Figure 4a) allows for a manageable number of commands per target below 10 even for $A < 50\%$, while in its absence (green for CA) the number of required commands raises dramatically below 90% accuracy. This explains why almost all failed or skipped tasks occurred in condition CA .

On the other hand, the similar average typing speeds between $hBCI+CA$ and CA (35.7 and 34.18 spc, respectively), are justified by the fact that subjects in the non-failed tasks of CA managed to maintain high command accuracies (green squares), in the range at which the number of commands is similar to the ones of $hBCI+CA$, Figure 4a. In the same figure, data-

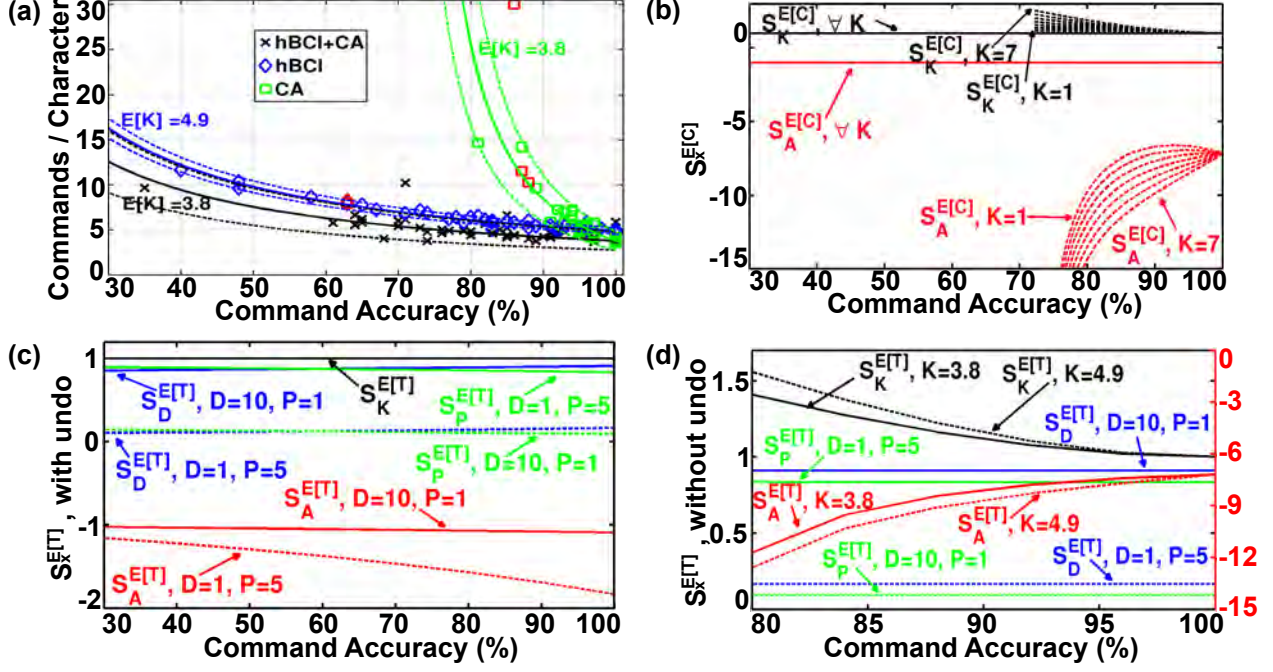


Figure 4: (a) Theoretical expectations $E[C]$ (solid lines for condition-dependent average K values and dashed lines for standard deviations, see Section 4.2) and experimentally measured C (points, shape and color-coded for the three protocol conditions in the legend) against command accuracy A . (b) Sensitivity functions $S_x^{E[C]}$, $x = A$ (red) and $x = K$ (black). Solid lines with and dashed lines without “undo”. (c)-(d): Sensitivity functions $S_x^{E[T]}$, $x = A$ (red), $x = K$ (black), $x = D$ (blue) and $x = P$ (green) with (c) and without “undo” (d). Solid and dashed lines for nominal point combinations (within the variable ranges measured in our experiments) that produce highest and lowest sensitivity magnitudes, as signified in the legends. $S_A^{E[T]}$ in figure (d) corresponds to the right Y-axis (red), all other sensitivity functions in this figure correspond to the left Y-axis (black).

compression in BrainTree is shown to account for around 1–2 less commands per character (black vs blue lines, depending on A), thus it is normal that the benefits of the *hBCI+CA* over the *hBCI* condition are mostly seen in the average typing speed.

4.3. Sensitivity analysis

Solid and generalizable conclusions on usability and efficiency are drawn by means of the relative importance of BCI-related factors, command accuracy A and delivery time D , as well as control paradigm-related factors K , H and P (as defined in Section 3) on the expectations of target variables C and T . The importance of these factors is evaluated through relative sensitivity functions $S_y^x|_p = \frac{\partial y}{\partial x}|_p \frac{x_p}{y_p} \approx \frac{\% \text{ change in } y}{\% \text{ change in } x}$ for nominal points

p . The magnitude $|S_x^y|$ reveals the impact of x on y , a positive sign denotes that y increases when x increases from its value at p , while a negative sign the opposite. Relative sensitivities $S_x^{E[C]}$ and $S_x^{E[T]}$ are presented in Figures 4b and 4c-d, respectively, against accuracy A . When the sensitivity of an investigated variable depends also on additional variables identified in our model, nominal points are chosen and shown by the figure’s legends to reflect (highest and lowest) sensitivity extrema of the variable in question. A wide range of possible (as measured in our experiments) nominal points is hence evaluated ($A \in [30, 100]$, $D \in [1, 10]$, $K \in [1, 7]$, $P \in [1, 5]$).

Command accuracy A is shown to be the most important factor in code-based applications; when “undo” is available (solid red lines) $|S_A^{E[C]}|$ is 1 (Figure 4b) and $|S_A^{E[T]}|$ ranges from 1 to around 2 (for high P and low D , Figure 4c), thus a 10% accuracy drop leads to a 10% increase of needed commands and 10-20% decrease of typing speed. Its importance raises dramatically when “undo” error-handling is not available (dashed red line in Figure 4b), where $|S_A^{E[C]}|$ starts around 7 and converges to infinity at the limit $\sqrt[M]{0.5}$, its slope increasing with increasing K (and M , not shown here). $S_A^{E[T]}$ has a similar trend (solid red line for $E[K] = 3.8$, dashed red for $E[K] = 4.9$ in Figure 4d, corresponding to the right Y-axis). A drop of accuracy from 100 to 90% thus leads to almost double number of commands and time needed for a selection ($\approx 100\%$ increase of $E[C], E[T]$ since $S_A^{E[C]}, S_A^{E[T]} \approx -10$, verified experimentally by the green squares of Figure 4a). The absence of “undo” therefore imposes a theoretical minimum requirement on accuracy, and also a practical one around $A = 80\%$ since subjects can anyway hardly sustain a required number of commands higher than $t_c \simeq 15$ (failed tasks in Figure 4a) or selection times higher than $t_t \simeq 1$ minute, Figure 3b.

The dependence of the slopes of $S_A^{E[C]}$ (Figure 4b) and $S_A^{E[T]}$ (Figure 4d) on M when “undo” is unavailable justifies the existence of many “delete” targets or other mechanisms to reduce the commands needed to reach a “delete” target, so that the strict theoretical as well as the practical accuracy requirement is reduced. In BrainTree, a high probability of 0.2 was forced for “backspace” to fix $M = 2$ and thus the minimum theoretical accuracy requirement corresponds to $\sqrt[2]{0.5} = 71\%$. However, depending on the application, a reduction of M might result in increase of average codeword lengths K , hence this trade-off should be carefully accommodated. Overall, it is shown that a perfect “undo” operation has tremendous impact on the usability and efficiency of code-based applications through its effect on A .

$|S_K^{E[C]}|$ and $|S_K^{E[T]}|$ reveal the importance of data-compression mechanisms, as they operate by reducing the average codeword length K . The magnitude of the sensitivity functions of variable K is constant at 1 (independent from command accuracy A) when “undo” is available (solid black lines, Figures 4b-c), and thus equally important to accuracy for $E[C]$, since $|S_A^{E[C]}| = 1$ and comparable for $E[T]$, since $1 \leq |S_A^{E[T]}| \leq 2$. This practically means that data-compression compensates accuracy losses to a fair extent by an equal reduction of K : this is verified for $E[C]$ in the case of BrainTree in Figure 4a.

Command delivery time D and pause intervals P only affect T , where $S_D^{E[T]}$ (blue) and $S_P^{E[T]}$ (green) are shown to be fairly insignificant (close to 0.1, Figures 4c-d) for low values of either D (fast BCI) or P and only having a more significant impact (sensitivities approaching 0.8) when the BCI is particularly slow ($D \simeq 10$ sec) or the pause too large ($P \geq 5$ sec), situations that rarely apply in practice since MI BCIs tend to be relatively fast (delivery time medians around 2–6 sec, Figure 3d) and large pause intervals are usually not necessary. These observations have important implications on a code-based application’s design. First, we should adjust all parameters that simultaneously affect command accuracy and delivery time (like decision thresholds, integration smoothing) with values that favour accuracy. Second, the length of pause intervals should be reduced as much as the user comfort permits. In case of BrainTree, reducing P from 3 seconds (used in our experiments) to 1 (which is sufficient for users to identify the next needed transition) revise our typing speed expectations from 35.7 spc (1.68 cpm) to 24.5 spc (2.45 cpm) as well as the best performance expectations from 16.5 spc (3.6 cpm) to 9 spc (6.6 cpm).

5. Discussion and conclusions

The main outcome of our study is the demonstrated extensive usability (100% flawless word completion rate for both user categories when all features are enabled) and competitive efficiency (1.7 cpm average and 3.6 cpm maximum user typing speed) of an MI BCI speller by 6 disabled and 10 able-bodied users, in real-world conditions and without requiring any re-calibration of the BCI classifier before spelling.

We argue that this result can be partially attributed to the absence of timing-based features in our speller. In our initial evaluation, 2 able-bodied users (H13-H14) and 3 end-users (E1-E3), using the same MI BCI, failed to use our timing-based speller [36] based on the scanning mode (Section 1), despite demonstrating more than adequate BCI performances in the training phase. E1 and E3 quit all attempted tasks, while E2 managed to complete certain tasks at an average speed of 1.2 cpm. Another two able-bodied users (H11 and H12) reached 1.82 and 1.1 cpm, respectively, after relaxing the requirement for error-free completion. Note that end-user E3 subsequently completed successfully the BrainTree evaluation. All spelling attempts showcased the difficulty to comply with the required fast transitions or avoid false positives by staying idle. Similar difficulties were reported in [11] and discussed in [12]. Although it might be possible to cope with these shortcomings with appropriate training (not included in conventional MI BCI training protocols, where only command accuracy and delivery speed are tested) we have shown that code-based paradigms, imposing low requirements for BCI accuracy and speed, can be a usable and simple solution for end-users.

Indeed, our typing speed results (in combination with a reduction of the pausing interval to 1 s) would yield a mean performance of 2.45 cpm (maximum performance of 6.6 cpm).

This performance is similar to that reported for timing-based interfaces ([11, 12]), other code-based spellers ([41]), and even competitive to P300 spellers, which tend on average to range within 5-10 cpm ([8]). We have thus also demonstrated that, despite still inferior in terms of efficiency to P300, the BrainTree speller can be an attractive alternative for end-users who either cannot operate a P300 speller for some reason (e.g. due to small P300 amplitudes or eye-movement impairments) or because they subjectively valued more the spontaneity of a self-paced paradigm.

Comparison between the two user categories reveals a slight, statistically insignificant superiority of able-bodied users in terms of command accuracy (91% median accuracy for end-users and 94% for able-bodied users, Figure 3c), also reflected in the similar task completion rates achieved by the two user groups in Figure 3a for condition *hBCI+CA*. On the other hand, a statistically significant difference at the 5% confidence interval is found regarding BCI command delivery times (medians of 4.1 sec for end-users against 2.9 for able-bodied users, Figure 3d) and, consequently, also typing speed in the *hBCI+CA* condition (average of 1.47 cpm for end-users against 1.84 cpm for able-bodied users, Section 4.1). However, end-users E7 and E8 were as fast in typing as the best able-bodied users. Although we cannot exclude the possibility that the slightly compromised efficiency of end-users E3–E6 is attributed to their medical condition, this effect could also be related to the limited BCI training they have received in comparison to the majority of able-bodied participants. Most importantly, the longer delivery times for end-users E3–E6 had absolutely no impact on their ability to flawlessly convey messages through our MI speller.

We have further presented and validated a generic theoretical model of code-based spellers and applications in general, establishing a methodology for analysing the minimum BCI control requirements and making performance predictions. Our model has also assisted in quantifying the expected benefits in a code-based application by means of data-compression HCI techniques and “undo” error-handling. Data-compression has been shown to significantly increase typing speed by reducing the average codeword length. Reducing the target set size (where applicable) or increasing the recognized mental states (only advisable if command accuracy is not compromised) can also be used towards the same goal.

Additionally, we have shown that an overall command accuracy of 80% seems to be practically necessary for creating a usable purely BCI-actuated code-based application, thus we have revised upwards the 70% minimum accuracy requirement suggested in [42] and frequently used as a rule of thumb thereafter (also in our study as an inclusion criterion), a finding in line with [39]. Yet, it is also demonstrated that “undo” error-handling greatly relaxes this requirement when implemented with a reliable input channel (EMG in our case). It is worth to note that although 4 end-users did not test the purely BCI-actuated spelling, by means of the theoretical accuracy requirement (which has been experimentally verified) it can be safely assumed that end-users E3 and E6, and probably also E4, should be able to spell without hybrid error correction, given that they could maintain the median accuracies

achieved in the other experimental conditions (see Figure 3c).

The theoretically established importance of “undo” on usability, efficiency and minimum accuracy requirements is another major outcome of our study, also addressed in [41]. However, we show that BCI command accuracy is the most important factor even in its presence. This fact justifies error-handling implementation through a hybrid BCI approach, because the addition of a third BCI class is likely to reduce the overall BCI accuracy. In other words, since the latter has been shown to be the most crucial factor for efficient and effective code-based applications, a hybrid implementation for corrections should be preferred to an additional MI task, at least when the end-user’s residual capabilities permit it.

Our particular EMG-based implementation for the “undo” functionality proved to be extremely reliable and its selection was based on our end-user pathological profiles (user-centered approach), since all of them maintained some usable residual muscular activity. However, for most of the end-users this muscular activity was largely inadequate for sustained binary control. We hereby focus on the importance of an “undo” command in code-based interfaces rather than the means to implement it; the latter can be any custom “switch” the user has control over (e.g. sip-and-puff, eye-blinks, etc.). Additionally, even for those participants who maintained adequate muscle residual, the evaluation of the MI BCI as an alternative control modality deserves its own merit; for instance, end-users might prefer it over muscle-based or other HCI interfaces or consider it as a useful alternative and/or a future option in case they suffer from degenerative diseases. Furthermore, although the end-users of this study would probably be more efficient with other HCI (e.g. eye-trackers), the benchmarking work and the conclusions reached hereby are valuable for pushing MI BCI spelling beyond current state-of-the-art.

The use of our EMG hybrid input channel is addressed to end-users with enough muscle residual to operate it reliably, but who cannot employ it too often due to the induced fatigue that it would generate. In case of imperfect hybrid “undo” command our model can be easily extended to account for this modification. The ratio $E[C_{undo}]/E[C] = 1 - A$, with accuracies mostly ranging from 80% to 90%, shows that EMG-based corrections need not be employed more than once every 5–10 BCI commands and thus complying with this infrequent use requirement.

Paired t-tests (5% confidence interval) between the BCI command accuracies (the most important factor identified here) for words spelled in consecutive conditions, individually for each subject, reveal that only subject E8 shows a statistical significant improvement over time. Rejecting the null hypotheses can only be attributed to learning effects, since conditions have been executed in increasing difficulty for the subjects. While learning effects cannot be universally observed, performing the same t-test over all subjects illustrates statistically significant accuracy improvements between the last two conditions (3.82%, $p=0.0107$) and between the first and the third condition (6.21%, $p=0.0011$), but none between the first two conditions. Weak learning effects hence seem to occur during spelling. At this point, it

is interesting to see how many subjects who successfully completed the hBCI and CA (2nd and 3rd) conditions would have failed if those conditions had been executed at the beginning of the experiment, before any learning effect. The hBCI condition should not be affected, since no improvement is shown between the first two conditions. For the CA (and toughest) condition, if we remove the 6.21% average improvement, command accuracy for all words remains above the theoretical requirement of 80% accuracy. Thus, our usability results regarding the CA condition should not have been affected either. It is also worth noting that this learning effect is a positive outcome, as our MI BCI speller facilitates learning with experience. This implies that the mental workload imposed by the system is manageable. The learning effect also strengthens the superiority of the full hBCI+CA condition since, despite being the first to be tested, yielded 100% task completion for all subjects (Figure 3a).

Concluding, we have hereby presented the first MI BCI spelling, end-user evaluation study involving a substantial number of end-users, proving the extensive usability and competitive efficiency of the proposed code-based MI speller. We have also provided the theoretical tools for usability and efficiency analysis of code-based applications in general. Our future work entails exploring BCI adaptation methods to cope with the intense command accuracy and delivery speed fluctuations during spelling, which still pose a major obstacle towards transferring BCI spelling technology out of the lab, into the real world.

Acknowledgments

The authors would like to thank N. Pattaroni, H. Dimassi, V. Buhlmann, M. Rimondini and E. Pianarosa for their contributions in the clinical evaluation, as well as all end-users for their time and effort. This work was supported by the European ICT Programme Project FP7-224631 TOBI. This paper only reflects the authors views and funding agencies are not liable for any use that may be made of the information contained herein.

Appendix A. Model justification

$E[C] = K/A$ when $H = 1$ (“undo” available, first leg of Equation 1), is based on the fact that the number n of digit/command insertions until the correct one is forwarded is geometrically distributed with $E[n] = 1/A$ (and for the number of failures q until the first success, $E[q] = (1 - A)/A$), accuracy A reflecting the probability of correct insertion (independent of the command set size, and thus applicable for N-class BCIs). Multiplying with K gives the expected command overhead for the whole codeword.

In case $H = 0$ (only “delete” target available), all digits of the target’s codeword have to be correctly forwarded, an event with probability A^K , and accordingly the expected number of failures before the correct target is reached is $(1 - A^K)/A^K$. A correct target reaching generates exactly K commands, while, each failure generates a potential number of extra

commands G until the procedure for reaching “delete” can be launched, as well as a number of commands O until “delete” is reached once more than the failures on it (since each failure to reach the “delete” target will generate more erroneous codewords to be deleted). In BrainTree, G reflects the random descending to an erroneous leaf node before “backspace” can be reached at the next selection round and can be approximated by the average tree depth estimated to be ≈ 5.2 for the trees appearing in our protocol. The above reasoning justifies the general form of the second leg of Equation 1.

To conclude our justification an expectation on O should also be derived; since “delete” is a target like any other in the dictionary with length M , it has a probability of correct insertion A^M , and thus the expectation on the number of commands when one attempts to reach it can be approximated by $MA^M + G(1 - A^M)$ considering that an attempt on “delete” could either succeed (with probability A^M , generating M commands), or fail (with probability $1 - A^M$, generating G commands), and in the latter case one more false codeword has been inserted and should be also deleted. It is a known result that the expected number of trials for a Bernoulli experiment with probability of success p until “successes” are more than the failures (in which case the user will find himself at the same point as before committing the first error) is $1/(2p - 1)$ if $p \geq 0.5$ and ∞ otherwise. In the case in question $p = A^M$, thus finally the expected attempts on “delete” will be $1/(2A^M - 1)$ and consequently an approximation on $E[O]$ can be derived as $E[O] = (1/(2A^M - 1))(MA^M + G(1 - A^M))$ or ∞ , depending on whether $A^M \geq 0.5 \Rightarrow A \geq \sqrt[M]{0.5}$. Substituting $E[O]$ in the second leg of Equation 1 gives the final derivation and justifies the distinction made on $E[C]$ based on the subject’s accuracy A .

References

- [1] Wolpaw J R, Birbaumer N, McFarland D J, Pfurtscheller G, and Vaughan T M. Brain-computer interfaces for communication and control. *Clin. Neurophysiol.*, 113(6):767–91, 2002.
- [2] Birbaumer N, Kübler A, Ghanayim N, Hinterberger T, Perelmouter J, Kaiser J, Iversen I, Kotchoubey B, Neumann N, and Flor H. The thought translation device (TTD) for completely paralyzed patients. *IEEE Trans. Rehabil. Eng.*, 8(2):190–3, 2000.
- [3] Donchin E, Spencer K M, and Wijesinghe R. The mental prosthesis: Assessing the speed of a P300-based brain-computer interface. *IEEE Trans. Rehabil. Eng.*, 8:174–9, 2000.
- [4] Sellers E W and Donchin E. A P300-based brain-computer interface: Initial tests by ALS patients. *Clin. Neurophysiol.*, 117(3):538–48, 2006.
- [5] Obermaier B, Müller G R, and Pfurtscheller G. Virtual keyboard controlled by spontaneous EEG activity. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 11(4):422–6, 2003.
- [6] Millán J d R. Adaptive Brain Interfaces. *Communications of the ACM*, 46(3):74–80, 2003.
- [7] Birch G E, Bozorgzadeh Z, and Mason S G. Initial on-line evaluations of the LF-ASD brain-computer interface with able-bodied and spinal-cord subjects using imagined voluntary motor potentials. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 10(4):219–24, 2002.
- [8] Cecotti H. Spelling with non-invasive Brain-Computer Interfaces - Current and future trends. *Journal of Physiology-Paris*, 105(1-3):106–114, 2011.

- [9] Mak J N, Arbel Y, Minett J W, McCane L M, Yuksel B, Ryan D, Thompson D, Bianchi L, and Erdogmus D. Optimizing the P300-based brain-computer interface: current status, limitations and future directions. *J. Neural Eng.*, 8(2):25–3, 2011.
- [10] Fazel-Rezai R, Allison B Z, Guger C, Sellers E W, Kleih S C, and Kübler A. P300 brain computer interface: current challenges and emerging trends. *Front. Neuroeng.*, 5(14), 2012.
- [11] Scherer R, Müller G R, Neuper C, Graimann B, and Pfurtscheller G. An asynchronously controlled EEG-based virtual keyboard: improvement of the spelling rate. *IEEE Trans. Biomed. Eng.*, 51(6):979–84, 2004.
- [12] Williamson J, Murray-Smith R, Blankertz B, Krauledat M, and Müller K R. Designing for uncertain, asymmetric control: Interaction design for brain-computer interfaces. *Int. J. Hum.-Comput. Stud.*, 67(10):827–41, 2009.
- [13] Volosyak I, Valbuena D, Luth T, Malechka T, and Gräser A. BCI Demographics II: How Many (and What Kinds of) People Can Use a High-Frequency SSVEP BCI? *IEEE Trans. Neural Syst. Rehabil. Eng.*, 19(3):232–39, 2011.
- [14] Treder M and Blankertz B. (C)overt attention and visual speller design in an ERP-based brain-computer interface. *Behav. Brain Funct.*, 6(1):28, 2010.
- [15] Orhan U, Erdogmus D, Roark B, Purwar S, Hild K E, Oken B, Nezamfar H, and Fried-Oken M. Fusion with language models improves spelling accuracy for ERP-based brain computer interface spellers. In *Conf. Proc. IEEE Eng. Med. Biol. Soc. 2011*, pages 5774–7, 2011.
- [16] Sellers E W, Kübler A, and Donchin E. Brain-computer interface research at the university of south Florida cognitive psychophysiology laboratory: the P300 speller. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 14(2):221–4, 2006.
- [17] Kübler A and Birbaumer N. Brain-computer interfaces and communication in paralysis: Extinction of goal directed thinking in completely paralysed patients? *Clin. Neurophysiol.*, 119(11):2658–66, 2008.
- [18] Li Y, Nam C S, Shadden B B, and Johnson S L. A P300-Based Brain-Computer Interface: Effects of Interface Type and Screen Size. *Int. J. Hum. Comput. Inter.*, 27(1):52–68, 2010.
- [19] Nijboer F, Sellers E W, Mellinger J, Jordan M A, Matuz T, Furdea A, Halder A, Mochty U, Krusienski D J, Vaughan T M, Wolpaw J R, Birbaumer N, and Kübler A. A P300-based brain-computer interface for people with amyotrophic lateral sclerosis. *Clin. Neurophysiol.*, 119(8):1909–16, 2008.
- [20] Hoffmann U, Vesin J-M, Ebrahimi T, and Diserens K. An efficient P300-based brain-computer interface for disabled subjects. *J. Neurosci. Methods*, 167(1):115–25, 2008.
- [21] Kübler A, Furdea A, Halder S, Hammer E M, Nijboer F, and Kotchoubey B. A Brain-Computer Interface Controlled Auditory Event-Related Potential (P300) Spelling System for Locked-In Patients. *Ann. N.Y. Acad. Sci.*, 1157(1):90–100, 2009.
- [22] Townsend G, LaPallo B K, Boulay C B, Krusienski D J, Frye G E and Hauser C K, Schwartz N E, Vaughan T M, Wolpaw J R, and Sellers E W. A novel P300-based brain-computer interface stimulus presentation paradigm: Moving beyond rows and columns. *Clin. Neurophysiol.*, 121(7):1109–20, 2010.
- [23] Sellers E W, Vaughan T M, and Wolpaw J R. A brain-computer interface for long-term independent home use. *Amyotroph. Lateral Sc.*, 11(5):449–55, 2010.
- [24] Birbaumer N, Ghanayim N, Hinterberger T, Iversen I, Kotchoubey B, Kübler A., Perelmouter J, Taub E, and Flor H. A Spelling Device for the Paralyzed. *Nature*, 398, 1999.
- [25] Orhan U, Hild K E, Erdogmus D, Roark B, Oken B, and Fried-Oken M. RSVP keyboard: An EEG based typing interface. In *IEEE Inter. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2012*, pages 645–8.
- [26] Brunner P, Ritaccio A L, Emrich J F, Bischof H, and Schalk G. Rapid communication with a P300 matrix speller using electrocorticographic signals (ECoG). *Front. Neurosci.*, 5(5), 2011.
- [27] Shih J and Krusienski D. Signals from intraventricular depth electrodes can control a brain-computer interface. *J. Neurosci. Methods*, 203(2):311–4, 2012.

- [28] Guger C, Edlinger G, Harkam W, Niedermayer I, and Pfurtscheller G. How many people are able to operate an EEG-based brain-computer interface (BCI)? *IEEE Trans. Neural Syst. Rehabil. Eng.*, 11(2):145–7, 2003.
- [29] Guger C, Daban S, Sellers E W, Holzner C, Krausz G, Carabalona R, Gramatica F, and Edlinger G. How many people are able to control a P300-based brain-computer interface (BCI)? *Neurosci. Lett.*, 462(1):94–8, 2009.
- [30] F. Galán, M. Nuttin, E. Lew, P.W. Ferrez, G. Vanacker, J. Philips, and J.d.R. Millán. A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots. *Clin. Neurophysiol.*, 119:2159–2169, 2008.
- [31] Zhang H, Guan C, and Wang C. Asynchronous P300-Based Brain-Computer Interfaces: A Computational Approach With Statistical Models. *IEEE T BIO-MED ENG*, 55(6):1754–63, 2008.
- [32] Aloise F, Schettini F, Aricò P, Leotta F, Salinari S, Mattia D, Babiloni F, and Cincotti F. P300-based braincomputer interface for environmental control: an asynchronous approach. *J Neural Eng*, 8(2):025025, 2011.
- [33] Panicker R C, Puthusserypady S, and Ying S. An Asynchronous P300 BCI With SSVEP-Based Control State Detection. *IEEE T BIO-MED ENG*, 58(6):1781–88, 2011.
- [34] H Gürkök. *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*. PhD thesis, University of Twente, Enschede, September 2012.
- [35] Friedrich E V C, Neuper C, and Scherer R. Whatever Works: A Systematic User-Centered Training Protocol to Optimize Brain-Computer Interfacing Individually. *PLoS ONE*, 8(9):e76214, 09 2013.
- [36] Perdikis S, Leeb R, Liboni N, Guigliemma C, and Millán J d R. BCI for Augmenting Communication Capabilities of Disabled People. In *Proc. of TOBI Workshop, Graz, Austria*, page 17, 2010.
- [37] Müller-Putz G R, Breitwieser C, Cincotti F, Leeb R, Schreuder M, Leotta F, Tavella M, Bianchi L, Kreiling A, Ramsay A, Rohm M, Sagebaum M, Tonin L, Neuper C, and Millán J d R. Tools for Brain-Computer Interaction: a General Concept for a Hybrid BCI (hBCI). *Front. Neuroinform.*, 5(30), 2011.
- [38] Pfurtscheller G, Allison B Z, Bauernfeind G, Brunner C, Solis-Escalante T, Scherer R, Zander T O, Müller-Putz G, Neuper C, and Birbaumer N. The hybrid BCI. *Front. Neurosci.*, 4(3), 2010.
- [39] Leeb R, Perdikis S, Tonin L, Biasiucci A, Tavella M, Creatura M, Molina A, Al-Khodairy A, Carlson T, and Millán J d R. Transferring braincomputer interfaces beyond the laboratory: Successful application control for motor-disabled users. *Artificial Intelligence in Medicine*, 59(2):121–132, 2013.
- [40] Hu T C and Tucker A C. Optimal Computer Search Trees and Variable-Length Alphabetical Codes. *SIAM J. Appl. Math.*, 21(4):514–32, 1971.
- [41] D’Albis T, Blatt R, Tedesco R, Sbattella L, and Matteucci M. A predictive speller controlled by a brain-computer interface based on motor imagery. *ACM Trans. Comput.-Hum. Interact.*, 19(3):20, 2012.
- [42] Kübler A, Neumann N, Wilhelm B, Hinterberger T, and Birbaumer N. Predictability of Brain-Computer Communication. *J. Psychophysiol.*, 18(2-3):121–9, 2004.