

Adaptive Brain-Computer Interface Systems For
Communication in People with Severe
Neuromuscular Disabilities

by

Boyla O. Mainsah

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

Leslie Collins, Supervisor

Galen Reeves

Loren W. Nolte

Stacy L. Tantum

Michael Gehm

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University

2016

ABSTRACT

Adaptive Brain-Computer Interface Systems For
Communication in People with Severe Neuromuscular
Disabilities

by

Boyla O. Mainsah

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

Leslie Collins, Supervisor

Galen Reeves

Loren W. Nolte

Stacy L. Tantum

Michael Gehm

An abstract of a dissertation submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in the Department of Electrical and Computer
Engineering
in the Graduate School of Duke University
2016

Copyright © 2016 by Boyla O. Mainsah
All rights reserved except the rights granted by the
Creative Commons Attribution-Noncommercial Licence

Abstract

Brain-computer interfaces (BCI) have the potential to restore communication or control abilities in individuals with severe neuromuscular limitations, such as those with amyotrophic lateral sclerosis (ALS). The role of a BCI is to extract and decode relevant information that conveys a user's intent directly from brain electro-physiological signals and translate this information into executable commands to control external devices. However, the BCI decision-making process is error-prone due to noisy electro-physiological data, representing the classic problem of efficiently transmitting and receiving information via a noisy communication channel.

This research focuses on P300-based BCIs which rely predominantly on event-related potentials (ERP) that are elicited as a function of a user's uncertainty regarding stimulus events, in either an acoustic or a visual oddball recognition task. The P300-based BCI system enables users to communicate messages from a set of choices by selecting a target character or icon that conveys a desired intent or action. P300-based BCIs have been widely researched as a communication alternative, especially in individuals with ALS who represent a target BCI user population. For the P300-based BCI, repeated data measurements are required to enhance the low signal-to-noise ratio of the elicited ERPs embedded in electroencephalography (EEG) data, in order to improve the accuracy of the target character estimation process. As a result, BCIs have relatively slower speeds when compared to other commercial assistive communication devices, and this limits BCI adoption by their target user

population. The goal of this research is to develop algorithms that take into account the physical limitations of the target BCI population to improve the efficiency of ERP-based spellers for real-world communication.

In this work, it is hypothesised that building adaptive capabilities into the BCI framework can potentially give the BCI system the flexibility to improve performance by adjusting system parameters in response to changing user inputs. The research in this work addresses three potential areas for improvement within the P300 speller framework: information optimisation, target character estimation and error correction. The visual interface and its operation control the method by which the ERPs are elicited through the presentation of stimulus events. The parameters of the stimulus presentation paradigm can be modified to modulate and enhance the elicited ERPs. A new stimulus presentation paradigm is developed in order to maximise the information content that is presented to the user by tuning stimulus paradigm parameters to positively affect performance. Internally, the BCI system determines the amount of data to collect and the method by which these data are processed to estimate the user's target character. Algorithms that exploit language information are developed to enhance the target character estimation process and to correct erroneous BCI selections. In addition, a new model-based method to predict BCI performance is developed, an approach which is independent of stimulus presentation paradigm and accounts for dynamic data collection. The studies presented in this work provide evidence that the proposed methods for incorporating adaptive strategies in the three areas have the potential to significantly improve BCI communication rates, and the proposed method for predicting BCI performance provides a reliable means to pre-assess BCI performance without extensive online testing.

Contents

Abstract	iv
List of Tables	x
List of Figures	xi
List of Abbreviations and Symbols	xiv
Acknowledgements	xvii
1 Introduction	1
2 Background	7
2.1 P300 Speller Operation	7
2.2 BCI Performance Evaluation	13
2.2.1 BCI Performance Prediction	14
2.2.2 Projected Accuracy for the CMA Static Stopping Algorithm	17
2.3 Information Optimisation	18
2.3.1 Physiological Effects	18
2.3.2 Stimulus Presentation Paradigms	19
2.3.3 Communication under Noisy Conditions	22
2.3.4 The P300 Speller Communication Channel	25
2.4 Target Character Estimation	29
2.4.1 Dynamic Data Collection	29
2.4.2 Statistical Language Models	30

2.5	Bayesian Dynamic Stopping	33
2.5.1	Algorithm Implementation	34
2.5.2	Inclusion of a Language Model	37
2.6	BCI Error Correction	41
2.6.1	Active User Correction	41
2.6.2	ErrP-based Correction	42
2.6.3	Language-based Correction	43
2.7	Summary	45
3	Predicting BCI Performance with the Classifier Detectability Index	47
3.1	Theoretical Model	48
3.1.1	The Bayesian Probability Random Variable	49
3.1.2	The Likelihood Ratio Test	50
3.1.3	Deriving Performance Estimates for the Bayesian Dynamic Stopping Algorithm	56
3.2	Methods	62
3.2.1	Analytical Estimation	62
3.2.2	Monte Carlo Estimation	68
3.3	Results	69
3.3.1	Verification of the Theoretical Model	69
3.3.2	Analyses with BCI Datasets	76
3.4	Discussion	82
4	Controlled Stimulus Selection	84
4.1	The Performance-based Paradigm	85
4.1.1	P300 Channel with Memory	88
4.1.2	Performance-based Parameters	90
4.1.3	Codebook Development Algorithm	93

4.2	Online Codebook Comparison Study	97
4.2.1	Methods	97
4.2.2	Results	98
4.3	Discussion	101
5	Increasing Language Model Complexity in the Target Character Estimation Algorithm	103
5.1	Higher Order Language Models	104
5.1.1	Methods	105
5.1.2	Results	108
5.2	Discussion	112
6	Error Correction in P300 Spellers	114
6.1	Error Information in P300 Classifier Outputs	115
6.2	Preliminary Analysis of Error-Correction Mechanisms	117
6.2.1	Methods	117
6.2.2	Results	120
6.3	Comparison of Spelling Correction Methods	123
6.3.1	Methods	124
6.3.2	Results	129
6.4	Discussion	132
7	Conclusions and Future Work	135
A	Random Sums	143
B	Random Inequalities	145
C	The Maximum of M Random Variables	147
D	Truncated Sequential Likelihood Ratio Test	149
E	Character Probability Initialisation in the Forward Algorithm	151

Bibliography	152
Biography	164

List of Tables

6.1	Statistical pairwise comparison between correction algorithms: Character performance	132
6.2	Statistical pairwise comparison between correction algorithms: Word performance	132

List of Figures

2.1	Electroencephalography responses elicited in an oddball recognition task.	8
2.2	The P300 speller components.	9
2.3	EEG electrode array for BCI signal processing.	10
2.4	Variation in the observed accuracy, A_{obs} , estimated from N characters, given an underlying system accuracy, A_{true}	15
2.5	Effect of target-to-target interval on the probability density function of the target P300 classifier scores.	20
2.6	A binary symmetric channel	22
2.7	Schematic of a communication system	24
2.8	Illustration of the P300 speller encoding process for the row-column paradigm	26
2.9	Illustration of the modulation process in an oddball recognition task	27
2.10	The character probability matrix for a letter bi-gram language model.	32
2.11	Bayesian dynamic stopping algorithm for the P300 speller.	35
2.12	Evolution of character probabilities in Bayesian dynamic stopping	36
2.13	Comparison of performance measures of ALS participants using different data collection algorithms	39
3.1	Illustrative example of the class conditional probability distribution parameters used to determine the detectability index of two normal distributions.	55
3.2	Role of probability density functions during the probability update process in the Bayesian dynamic stopping algorithm	57

3.3	Accuracy vs. detectability index, with the Bayesian algorithm for a row-column paradigm with a 9×8 grid.	72
3.4	Expected stopping time vs. detectability index with the Bayesian dynamic stopping algorithm for a row-column paradigm with a 9×8 grid	73
3.5	Accuracy vs. detectability index with the Bayesian algorithm with fixed sample size for row-column and non-randomised checkerboard paradigms, both implemented with a 9×8 grid	75
3.6	Performance vs. detectability index, for the 9×8 row-column paradigm, obtained from simulating Bayesian dynamic stopping with participant EEG data	79
3.7	Performance vs. detectability index, obtained from online studies using Bayesian dynamic stopping with a 9×8 row-column paradigm	81
3.8	Performance vs. detectability index, obtained from online studies using Bayesian dynamic stopping with a 6×6 checkerboard paradigm	81
4.1	Performance comparison for different stimulus paradigms with (36,24)-codes: expected stopping time vs. accuracy	87
4.2	Comparison of the discriminability between target-to-target interval segregated classifier scores and the aggregate non-target or target classifier scores, of three stimulus paradigms: row-column, checkerboard and random paradigms.	92
4.3	Performance-based paradigm for a (36, 24)-code	95
4.4	Performance vs. detectability index for the row-column paradigm and performance-based paradigm based on simulations with synthetic data.	95
4.5	Online results comparing the row-column paradigm and the performance-based paradigm	99
4.6	Comparison of participant detectability indices of the test data for the row-column paradigm and performance-based paradigm.	100
4.7	Performance vs. detectability index for the codebook comparison study: row-column paradigm and performance-based paradigm	100
5.1	Offline performance of simulations of different dynamic stopping algorithms with language models	109

5.2	Online performance for the Bayesian dynamic stopping algorithms with different order language models.	112
6.1	Distribution of Bayesian character probabilities post-data collection.	116
6.2	ErrP corrective mechanism following P300 speller selection.	119
6.3	Effective number of P300 speller selections vs. probability of error of a P300 speller selection.	121
6.4	Effective number of P300 speller selections vs. probability of error of a P300 speller selection, with and without an ErrP cascade.	121
6.5	Selection of an optimal operating point for the ErrP classifier in a P300-ErrP cascade	122
6.6	Flowchart for proposed spelling correction in the P300 speller.	126
6.7	Participant results for P300 speller accuracy, with and without use of spelling correction algorithms.	130

List of Abbreviations and Symbols

Symbols

$f_X(x)$	Probability density function of the random variable X , evaluated at a value x .
$F_X(x)$	Cumulative distribution function of the random variable X , evaluated at a value x .
$\mathbf{1}_{\{\cdot\}}$	Indicator function which evaluates to 1 if condition in $\{\cdot\}$ is true, and 0 otherwise.
\hat{X}	Estimate of a variable, X .
$\bar{X}/\mathbb{E}[X]$	Mean or expected value of random variable, X
$\text{Var}[X]$	Variance of random variable, X

Abbreviations

AAC	Augmentative and Alternative Communication
ALS	Amyotrophic Lateral Sclerosis
ASN	Average Sample Number
BCI	Brain-Computer Interface
BSC	Binary Symmetric Channel
CBLE	Classifier-based Latency Estimation
CBP	Checkerboard Paradigm
CDF	Cumulative Distribution Function
CLLR	Cumulative Log-Likelihood Ratio
CMA	Cumulative Moving Average

d	Detectability Index
d^H	Hamming Distance
DS	Dynamic Stopping
DSLMM	Dynamic Stopping with Language Model
EEG	Electroencephalography
EST	Expected Stopping Time
ERP	Event-related Potential
ErrP	Error-related Potential
FSS	Fixed Sample Size
HMM	Hidden Markov Model
IID	Independent and Identically Distributed
KDE	Kernel Density Estimation
LI	Locked-in
LLR	Log-Likelihood Ratio
MLE	Maximum Likelihood Estimator
NAC	Non-alphabetic Character
OOV	Out-of-Vocabulary
PBP	Performance-based Paradigm
PDF	Probability Density Function
P_D	Probability of Detection
p_e	Probability of Error
P_{FA}	Probability of False Alarm
P_{th}	Probability Threshold
RCP	Row-column Paradigm
RIP	Random Isometric Property
RSVP	Rapid Serial Visual Presentation

SLRT	Sequential Likelihood Ratio Test
SNR	Signal-to-noise Ratio
SWLDA	Stepwise Linear Discriminant Analysis
TBR	Theoretical Bit Rate
TTI	Target-to-target Interval
TDS	Truncated Dynamic Stopping
UDS	Untruncated Dynamic Stopping

Acknowledgements

I would like to thank the various organisations who provided financing for my graduate education: Duke University Graduate School through the First Year Fellowship, the National Institute of Health, the National Institute on Deafness and Other Communication Disorders and the Kristina M. Johnson Fellowship.

I am grateful to all those who participated in my research studies as they helped provide the empirical evidence necessary to validate my research. To those with ALS, it is my hope that my research will contribute towards something meaningful in the quest to connect individuals to their loved ones and the outside world.

I would like to thank Drs. Eric Sellers and David Ryan at East Tennessee State University and Kevin Caves at Duke University for their collaboration in helping with my BCI experiments, from participant recruitment to research data collection.

I would like to thank all the members of my exam committees who approved all of my academic milestones: Drs. Rebecca Willett, Kishor Trivedi, Willian Joines, Michael Gehm, Stacy Tantum, Galen Reeves and Loren Nolte. Special thanks to Drs. Reeves and Nolte whose insight and research advice contributed to conceptualising and concretising a significant portion of my research.

I would especially like to thank the two people responsible for training me into the independent researcher that I am today, Drs. Leslie Collins and Chandra Throckmorton. Thank you for the encouragement and orientation, for recognising my capabilities and for always pushing me beyond my academic comfort zone. I will always

appreciate your constructive feedback and constant instruction of focusing on telling a compelling story about my research. To Leslie, I will always be grateful to you for accepting me to into your lab. I have appreciated your constant help and support and you going to the mat for me throughout my Ph.D. journey. To Sandy, I couldn't have asked for a better research advisor and mentor. You allowed me not only the freedom to explore my own research path, but also provided the guidance to ensure I didn't veer too far off course.

To all the members of the Duke University SSPACISS lab whom I have had the pleasure of working with during my Ph.D. journey. To past members, Drs. Peter Torrione, Kenneth Morton, Christopher Ratto, Sara Duran, Jill Desmond, Achut Manandhar and Patrick Wang. To the present members: the research staff, Drs. Mary Knox and Jordan Malof, and graduate students, Rayn Sakaguchi, Kenneth Colwell, Nicholas Czarnek, Dmitry Kalika, Kedar Prabhudesai, Daniël Reichman, Joseph Camilo, Jillian Clements, Rui Hou, John Bralich, Bohao Huang and Ja'lon Sisson. Thank you for creating an amicable and engaging work environment. I would like to thank Kenny for research advice and guidance, Ken for introducing me to BCIs and Kedar for the BCI2000 software tutorials to implement my work.

And finally, I would like to thank God for my family: to Mami and Baa for the gift of life, to my siblings, Yefon, Lenjo, Justice, Karl, Shamwun and Shubuka. I am grateful for your continuous unconditional love, emotional and financial support, especially throughout my Ph.D. journey.

This dissertation is dedicated to my sister, Yefon, whose encouragement and support allowed me to have a career in engineering.

1

Introduction

Brain-computer interfaces (BCI) have the potential to restore control or communication abilities in individuals whose severe neuromuscular limitations, e.g. due to neuronal diseases, stroke or traumatic brain injury, limit or preclude the use of most conventional assistive communication technologies [1]. BCIs operate by monitoring electro-physiological signals, invasively or non-invasively, from different areas of the brain, thereby bypassing the brain's traditional neuronal pathways of nerves and effector muscle organs. These signals are processed by the BCI in real time to extract features that help discern the user's intent and this information is translated into executable commands to control different types of devices, e.g. [2, 3, 4].

People with amyotrophic lateral sclerosis (ALS), commonly known as Lou Gehrig's disease, represent a target user population to whom BCI use can have a significant impact on their ability to communicate. ALS is a degenerative neurological disease that fatally attacks motor neurons that relay brain signals from the spinal cord to muscles, leading to a progressive loss of voluntary muscle control despite retention of cognitive abilities [5]. Currently, there is no cure for the disease. There are approximately 20,000-30,000 cases to date in the US, with an additional 5000 diagnosed each

year [6]. The rate of the progression of symptoms in ALS varies between individuals and this affects the type of communication method or augmentative and alternative communication (AAC) device that can be used, e.g. alternate keyboards, touch-screen devices, head or eye tracking systems [7, 8]. In the late stages of the disease, referred to as the *locked-in* (LI) stage, there is total voluntary muscle paralysis. At the LI stage, communication is achieved mainly by low-tech approaches such as manual alphabet/communication boards, or high-tech approaches such as eye-tracking systems. However, manual boards are slow to use and require another individual to facilitate communication and eye-tracking can become problematic when there is an inability to sustain controlled eye movements [9]. There is a need for a more independent means of communication that can be used by these individuals without as much supervision to significantly improve their quality of life.

Individuals with ALS still retain the cognitive ability to control BCIs [5]. BCIs can potentially provide a beneficial alternative to manual communication boards, or in the case where eye-tracking is not feasible [9]. The most commonly researched BCI for people with ALS is the P300-based BCI speller that relies predominantly on detecting event-related potentials (ERP) in electroencephalography (EEG) data [10]. The P300 speller system enables users to communicate messages from a set of choices by selecting a target character or icon that conveys a desired intent or action. The P300 speller has been shown to be a viable communication system for people with ALS with signal stability that allows for long-term use [7, 11, 12, 13, 14].

Transitioning the P300 speller from the research lab into the home is a key goal in BCI development that requires robust algorithms that improve P300 speller accuracy and communication rates to levels that are practical for independent day-to-day use. Most BCI systems used by people with disabilities are managed within a controlled research environment [15, 16], with a limited number of commercial systems available for home use, e.g. [17]. BCIs generally have slower selection times when compared

to commercial AAC devices, which limits adoption in their target user population.

Most research efforts to improve BCI spelling rates primarily involve offline analyses of EEG datasets or online testing in non-disabled study participants, with few studies involving participants with severe neuromuscular disabilities who actually represent the target BCI population [8, 18, 19, 20]. Since online studies with human participants are time consuming and expensive, offline analyses serve as a fast way to identify and develop a few promising algorithms for later online implementation and experimental results. However, real-time closed-loop online BCI use, with feedback, provides a significantly better measure of the performance of an algorithm as there is independence of training and test datasets and the user interacts and modifies behaviour as the BCI system conveys the user’s intent. Online studies in non-disabled individuals are usually performed due to easier participant recruitment (usually young and healthy adults) at universities where most BCI research laboratories are located. Results from healthy participants may not necessarily be applicable to the disabled population due to more variability in disease etiology or disability progression. In addition, most BCI algorithms are tested online using simple online tasks like single-word copy-spelling tasks, whereas more practical testing for real-world applications would involve free-spelling of phrases or sentences.

The focus of this BCI research is to develop algorithms that take into account the physical limitations of the target BCI population to improve the efficiency of ERP-based spellers for real-world communication. Given the non-stationarity of a user’s BCI performance, inter-user performance variability and the wide array of messages that can be communicated with a BCI system, it is hypothesised that building adaptive capabilities into the BCI decision-making algorithm provide an efficient way for improving BCI speller performance. The proposed research in this work will address three potential areas for improvement within the speller system: information optimisation, target character estimation and error correction. The ERP-based speller can

be considered to consist of three parts. There is a visual interface and its operation which controls the method by which the ERPs are elicited through the presentation of stimulus events. Internally, the system determines the amount of data to collect and the method by which these data are used to estimate the user's target character. And, finally, the selected target character is chosen and displayed to the user. It is hypothesised that each of these three aspects of the system has the potential for improvement.

In the case of the visual interface, typically the subsets of characters that are presented simultaneously to the user in a stimulus event are chosen *a priori* based on the stimulus presentation paradigm. These groups of characters are typically presented at random, usually with limited consideration of the effect of the stimulus presentation paradigm parameters, such the flash group composition, flash group presentation order, etc., on ERP elicitation. For example, given a P300 speller with a matrix layout for the user interface, an intuitive method is to group characters by the rows and columns of the matrix and present them in a random order. This is the basis for the row-column paradigm [21], which is the predominant stimulus presentation paradigm used in most BCI-speller studies [18]. In this research, a new stimulus paradigm will be developed based on principles from coding theory [22, 23] since the P300 speller can essentially be considered as a noisy communication channel. By characterising the parameters of the channel, it is hypothesised that better stimulus presentation patterns could be designed to increase the information content that is transmitted, thereby improving P300 speller performance.

The second research area that will be investigated with the goal of improving the performance of the overall system is the target character estimation process. ERP-based BCI spellers are primarily limited by their long character selection times (i.e. slow spelling speeds) as a result of the need to record multiple repetitions of data to enhance the low signal-to-noise ratio of ERPs embedded within noisy EEG data for

improved selection accuracy. Strong evidence exists in the literature that adaptive data collection, rather than the static collection of a predetermined amount of data, improves speller accuracy and communication rate [24, 25]. These adaptive algorithms rely on classifier outcomes to determine the amount of data to collect based on a threshold-based function, e.g. a probabilistic level of confidence in selecting the target character. It is hypothesised that the incorporation of language information into a dynamic data collection algorithm could further improve target character estimation by potentially reducing the amount of data necessary for accurate character estimation. In this research, *a priori* language information will be incorporated into the data collection algorithm via statistical language models to improve performance for real-world communication.

For practical reasons, a maximum data collection limit is usually imposed in order to avoid the possibility of a BCI system taking an unreasonably long time to make a decision. Usually the same data limit is imposed across users or the data limit is determined from a user's calibration data. However, depending on a user's performance level, the value of the data collection limit can have a significant impact on the maximum possible accuracy level that the system can achieve. In this work, a new method is developed that quantifies BCI performance by the discriminability of the BCI system's classifier. The proposed method can serve as a useful tool to pre-assess BCI performance in order to determine a suitable data collection limit that is required to achieve a desired accuracy level.

The final research area that will be investigated for improvement is correcting erroneous BCI decisions. Current P300 speller systems select the target character based on the internal target character estimation algorithm without an outside method to detect and correct erroneous BCI selections. Some studies in the literature have proposed error correction methods that rely on error-related potentials (ErrP) which are EEG signals that are elicited in response to a user observing er-

roneous actions or feedback [26]. In ErrP-based error correction algorithms, erroneous BCI selections are automatically deleted upon ErrP signal detection. However ErrP-based error correction mechanisms come at a cost in data and time and can be unreliable depending on the system operating conditions [27, 28]. Alternatively, most communication devices have various language-based tools for error correction, e.g. auto-correct, spell-check etc, by inference due to the predictability of language. In this research, a new spelling correction method is developed by exploiting information from the P300 speller target character estimation process for language-based spelling correction, thereby eliminating the need for an additional signal processing unit to perform ErrP-based error correction.

2

Background

Event-related potentials (ERP) are brain signals that are elicited in response to specific stimulus events. ERPs serve as good input control signals for BCI systems because they are transient and time-locked to the eliciting event. The P300-based BCI, initially developed by Farwell and Donchin [21], is based on the ability of a BCI system to elicit and detect ERPs called P300 signals. In their benchmark experiment, Farwell and Donchin demonstrated the feasibility of using a P300-based BCI for spelling with electroencephalography (EEG) data recorded from just a single electrode. Currently deployed P300 spellers still utilise the same basic BCI framework, with modifications in the electrode channel montage, the stimulus presentation paradigms and the signal processing strategies. This chapter describes background material of the P300 speller system used in this research study.

2.1 P300 Speller Operation

The P300-based BCI relies predominantly on ERPs that are elicited as a function of a user's uncertainty regarding stimulus events in either an acoustic or a visual

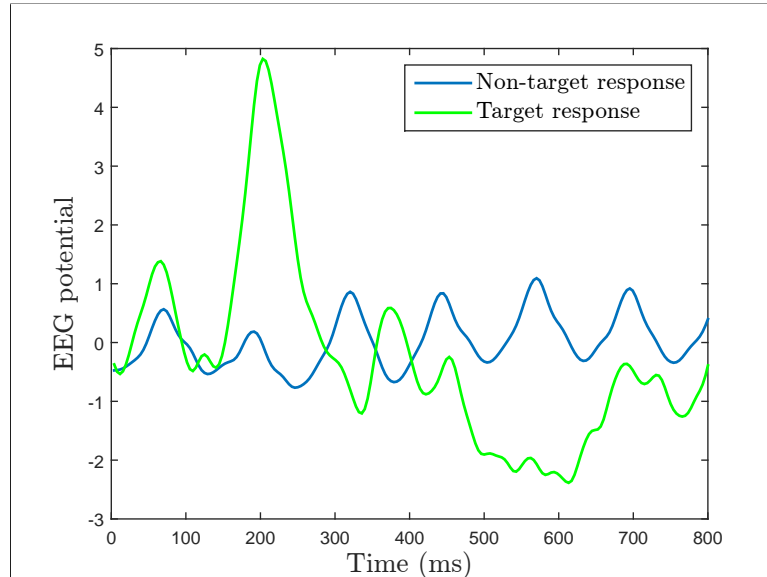


FIGURE 2.1: Electroencephalography (EEG) responses elicited in an oddball recognition task. A stimulus event occurs in the time interval 0 - 62.5 ms. The average EEG potentials elicited following non-target (blue) and target (green) stimulus events are shown. For the target response, a characteristic positive peak is observed within 200 - 500 ms from stimulus onset, termed the P300 signal.

oddball recognition task [10]. The random occurrence of a rare stimulus event within a sequence of stimulus events elicits a distinct ERP response that includes a large positive deflection called the P300 signal. The positive peak exhibits a variable latency, $\approx 200 - 500$ ms, attributed to the differences in information processing time required to classify each stimulus event [29]. Figure 2.1 shows EEG responses from a study participant that were obtained from averaging time windows of EEG data following target and non-target stimulus events. The *oddball* or target stimulus events, i.e. events that fall into the less frequent class, elicit a P300 signal while stimulus events that occur more frequently, the *background* or non-target stimuli, do not.

The P300 speller system consists of a signal acquisition unit with EEG electrodes and bio-amplifiers, a screen that displays character choices (alphanumeric, punctuation signs, keyboard commands etc.), and a computer system that runs the signal

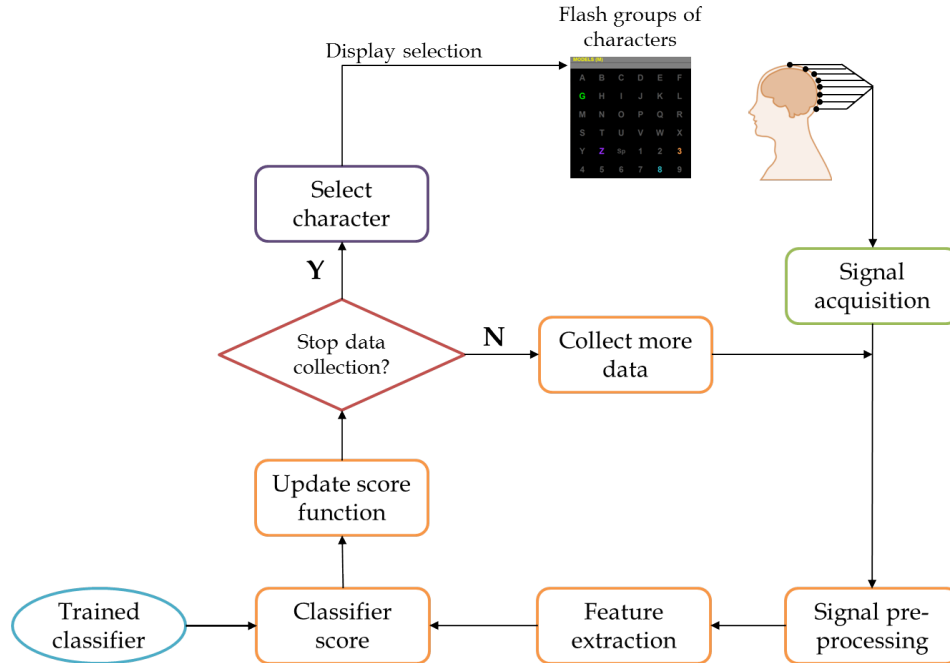


FIGURE 2.2: The P300 speller components. The user is presented with a grid of character choices. The user focuses on a desired or target character as groups of characters are randomly illuminated on the screen. External electrodes are used to measure EEG signals from the scalp, which are amplified, filtered and digitised for signal processing. Time sample blocks synchronised to stimulus onsets are used to extract feature vectors, which are scored with a trained classifier. After each stimulus event, the classifier score is used to update a character cumulative response function that quantifies how likely a character is the target character. After a certain amount of data collection, the character that maximises the cumulative response function is selected as the target character. The selected choice is presented as the user’s intended choice.

processing algorithm that translates the user’s intent into executable computer commands, as well as the speller’s operating protocol (stimulus event timing, feedback presentation, etc.) [5].

Figure 2.2 shows a schematic of the P300 speller system. Electrical signals for the P300 speller are acquired externally from the scalp via electroencephalography; a few studies have used the more invasive signal acquisition technique of electrocorticography, where an electrode array is placed below the skull over the cortex [30, 31, 32]. Since the P300 signal is predominantly distributed in the centro-parietal region of the scalp, most P300-based BCIs use midline electrodes, Cz, Fz and Pz, and a com-

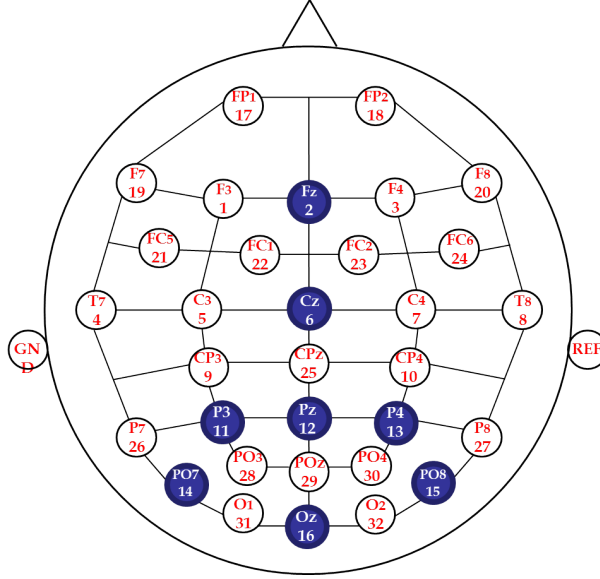


FIGURE 2.3: EEG electrode array for BCI signal processing. Electrodes highlighted in blue represent the set of electrodes used in this work.

bination of other surrounding electrodes [5]. In this work, the set of 8 electrodes, $\{Fz, Pz, Cz, P3, PO7, P4, PO8, Pz, Oz\}$, are used for signal processing (highlighted in blue in Figure 2.3), as this set has been shown to be generally applicable across users to improve performance [33]. EEG signals are amplified, band-limited, usually from 0.5-30Hz, and digitised for software processing [1]. Additional processing is often employed to minimise artifacts, e.g. a 60 Hz notch filter to remove electrical line noise.

To operate the P300 speller, a user is presented with a set of choices on a screen, denoted here as $\{C_m\}_{m=1}^M$ for an M -choice BCI. The user focuses on a desired or target character, C^* , as groups of characters, \mathcal{F}_t , are randomly illuminated on a screen: the illumination of the target character elicits a P300 ERP response. The row-column paradigm (RCP) [21] is the predominant stimulus paradigm used in most P300 spellers [18]. In the RCP, the character subsets are the rows and columns of a grid and the order of presentation of the flash groups is randomised without replacement. Following each stimulus event, EEG data from select electrodes are

processed to extract relevant information called features that in theory convey the user’s intent [1]. From the electrodes, EEG data obtained sliding time windows, synchronised with stimulus onsets, are used to extract feature vectors for signal processing. These extracted features are scored with a classifier which quantifies the EEG responses to distinguish between non-target and target stimulus events. The classifier is user-specific as it is trained based on EEG data of the current BCI user collected during a calibration run that is performed prior to online BCI use.

In the calibration run, the user performs a copy-spelling task with no classifier use or BCI feedback. In a copy-spelling task, the user is instructed by the BCI on which character in the grid to focus on. A truth label, $l \in \{0, 1\}$, is assigned to each feature vector, indicating the presence ($l_t = 1$) or absence ($l_t = 0$) of the target character in the associated flash group, \mathcal{F}_t . P300 signal detection lends itself to a binary classification problem where the goal is to decide whether EEG-based features are produced from target or non-target stimulus events, by mapping the high dimensional feature vector into a 1-dimensional space that maximises the separation between the two signal classes. A training dataset of feature vectors and corresponding truth labels, $\mathcal{T} = \{(\mathbf{f}_1, l_1), (\mathbf{f}_2, l_2), \dots, (\mathbf{f}_M, l_M)\}$, obtained from the calibration run is used to train the classifier.

For this work, the signal processing and classifier training approach developed by Krusienski *et al.* (2008) [33] will be used. At each electrode, EEG data obtained from a time window of 800ms is down-sampled from 256 Hz to 20 Hz using bin-averaging and the bin averages are concatenated across the 8 channels to obtain a feature vector, $\mathbf{f}^{1 \times 120}$. These labeled features are used to train a stepwise linear discriminant analysis (SWLDA) classifier, with weights $\mathbf{w}^{1 \times 120}$. The SWLDA method was chosen because it has been shown to result in higher classification rates across users when compared to other classification methods [34]. A classifier score, y , is generated from a dot product of the feature and the classifier weight vectors, $y = \mathbf{w} \mathbf{f}^\top$.

Single trial ERP detection after each stimulus event presentation is challenging given the low SNR of ERPs embedded in noisy EEG data [35]. As a result, the signal detection process is error-prone. For improved accuracy, character selection is made after averaging EEG data over multiple stimulus events to enhance the P300 ERP [36, 37]. In the current state-of-the-art method, the amount of data collection is fixed before online operation, termed *static data collection* or *static stopping* [21].

The classifier scores, $\mathbf{Y}_t = [y_1, y_2, \dots, y_t]$, are used to update a cumulative response function for each character according to its specific flash pattern. The cumulative response function, defined as $\Theta_m(t)$, determines how likely a character, C_m , is the target character at a time index, t . $\Theta_m(t)$ depends on the classifier responses and a character's presentation pattern. Within a data collection context, a *sequence* is a unit of data collection that consists of the number of defined flash groups in the base pattern of a given stimulus paradigm. For example, for the RCP with an $R \times C$ grid, 1 sequence = $R + C$ flashes. The data collection limit is usually quantified by the number of sequences. After data collection, the character that maximises the cumulative response function is selected as the user's intended target, $\hat{C} = \underset{m}{\operatorname{argmax}} \Theta_m(t)$, where \hat{C} is the estimate of the target character, C^* , as determined by the BCI.

The typical static stopping algorithm is evaluating the cumulative moving average (CMA) of classifier scores for each character [21]. Prior to data collection, all of the characters are assigned a zero score, $\psi_m(0) = 0$, $\forall m \in \{1, 2, \dots, M\}$, where $\psi_m(t)$ is the CMA of the classifier scores for character, C_m up to the t^{th} stimulus event. With each new stimulus flash, a character's cumulative response is updated according to:

$$\psi_m(T) = \frac{1}{f_m} \sum_{t=1}^T y_t \mathbf{1}_{\{C_m \in \mathcal{F}_t\}} \quad (2.1)$$

where f_m is the number of times character C_m is flashed in the time interval index 1 to T ; y_t is the classifier score obtained from the t^{th} stimulus flash; $\mathbf{1}_{\{C_m \in \mathcal{F}_t\}}$ is an indicator

function that is equal to 1 if $C_m \in \mathcal{F}_t$, and equal to 0 otherwise. Consequently, only the responses of characters present in the current stimulus event are updated. After a fixed amount of data collection, the character with the maximum average classifier score is selected as the user’s intended target character, $\hat{C} = \underset{m}{\operatorname{argmax}} \psi_m(T)$.

2.2 BCI Performance Evaluation

Various performance metrics have been proposed to assess the usability and benefit provided by BCI systems following system calibration [38, 39, 40]. A typical BCI evaluation protocol consists of three steps: (i) collection of EEG data to train the algorithm, (ii) algorithm parameter estimation from the training data (e.g. classifier selection, channel selection, etc.), and (iii) an online test which involves copy-spelling tasks using the BCI with the estimated parameters. Online validation is the gold-standard that is used to evaluate the closed-loop performance of a BCI algorithm, as the user interacts with the system and can modify behaviour in response to BCI feedback. The performance measures are typically estimated from data collected during the test phase of BCI evaluation.

Accuracy is a standard measure to evaluate a BCI’s performance and measures the probability that the BCI system correctly selects the user’s intended choice. It is usually determined from the ratio of the number of characters correctly selected by the BCI, N_c , to the number of characters a user intends to communicate, N . Selection speed is another performance measure that quantifies the time required to select a character. It can be a fixed or an average value, depending on whether the stopping criterion for data collection is static or dynamic, respectively. Some proposed performance measures to evaluate BCI performance rely on accuracy and selection speed, e.g. confusion matrix [41], bit rate [42], and utility [27].

Although online performance evaluation is important, it usually requires extensive

online testing which can be time consuming. As a result, in most BCI studies, performance estimates obtained from online BCI use rely on a low number of samples, $N \approx 10 - 40$. It is important to be aware of the variance of maximum likelihood estimators (MLE) [43]. Depending on the number of samples used, the estimated online performance may not necessarily reflect the underlying system performance if the estimate has a high variance.

For example, the observed accuracy, A_{obs} , calculated after online BCI use, follows the MLE of the parameter, p , of a binomial distribution, $n \sim B(N, p)$ [43], where N is the number of total observations, $p = A_{true}$, and A_{true} is the underlying system accuracy. Hence $A_{obs} = \hat{p}_{MLE} = N_c/N$, where N_c is the number of correctly selected characters or successes. \hat{p}_{MLE} is also a random variable, with mean, $\mathbb{E}[\hat{p}_{MLE}] = p$, and variance, $\text{Var}[\hat{p}_{MLE}] = p(1 - p)/N$. Figure 2.4 illustrates the possible variation that can be exhibited by A_{obs} based on the number of samples used for estimation. The variance of A_{obs} reduces with increased N . In addition, the resolution of the estimate also increases with N . While it is recommended that performance measures be obtained from a large number of samples [44, 45], this may not always be practical in real-world BCI experiment design. A method to infer a user’s long term performance without extensive online testing would be very useful to BCI users and developers.

2.2.1 BCI Performance Prediction

Predicting a user’s performance with a BCI is important for several reasons. For example, the predictions can be used to select parameters for the current stimulus paradigm (e.g. the amount of data to collect), to estimate the impact of switching to a new stimulus paradigm, or to avoid the frustrating scenario of a user attempting to use the system despite a low likelihood of success. Usually, performance can be predicted from cross-validation of the training data collected during the calibration

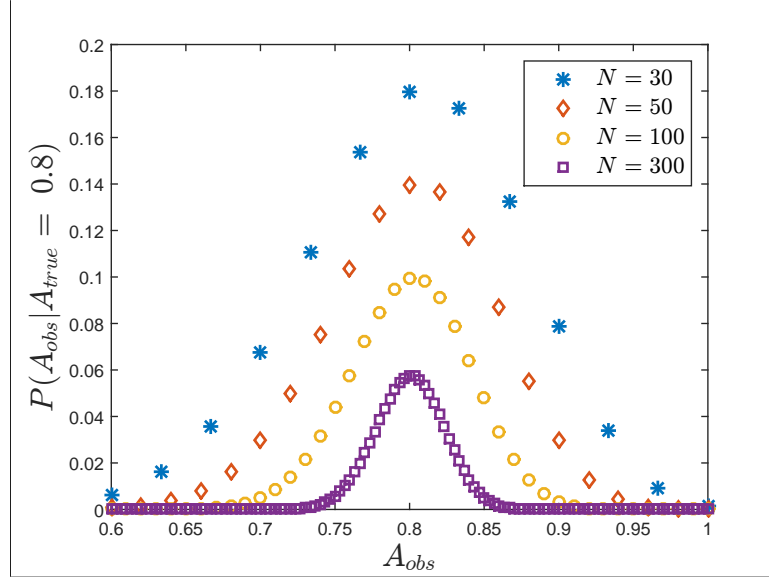


FIGURE 2.4: Variation in the observed accuracy, A_{obs} , estimated from N characters, given an underlying system accuracy, A_{true} . Displayed is the probability distribution of a binomial random variable, $n \sim B(N, p)$, where n corresponds to the number of correctly selected characters, $p = A_{true}$, and scaled such that $A_{obs} = n/N$. The x -axis is truncated (i.e. $A_{obs} \geq 0.6$), for visualisation purposes.

run. Some methods rely on estimates obtained empirically from a pool of users [35, 46, 47]. Thompson *et al.* (2013) [46] proposed a method to predict BCI accuracy based on classifier-based latency estimation (CBLE), a measure that quantifies a classifier’s sensitivity to ERP latency variability. However, methods that rely on data from a pool of users, such as the CBLE, require a large user database and are easily susceptible to user variability. For users from a population with disabilities, the varying progressions of their disease may result in significant differences between users.

In some studies, parameter selection for algorithm development involves a grid search over parameter values in order to maximise the performance of a user based on the training data [48, 49, 50]. Townsend *et al.* (2010) [50] proposed a method to select a data collection limit based on the amount of data that maximises the written symbol rate (WSR), a metric that represents the number of selections a participant

can correctly make in one minute, taking into account error correction. However, the maximum possible amount of data collection that can be suggested with a WSR-based method is the maximum amount used during calibration, whereas a higher amount of data collection might be required to achieve acceptable accuracy levels given a user’s performance level.

While performance predictions are usually made based on a user’s training data, these predictions may not always be consistent with performance observed in subsequent data collections. This is due to a potential shift in the data statistics between the two data collections. EEG data is non-stationary [51], and this can result in a mismatch of parameters developed from the training dataset and applied to subsequent data. Nonetheless, the training data does provide some useful prior information to ballpark a user’s performance level. Long-term performance can also be inferred post-hoc from a limited amount of data obtained from online BCI use via re-sampling techniques on the test dataset, such as bootstrapping [52], although there is still the issue of the non-stationarity of EEG signals.

It is hypothesised that methods for assessing long-term BCI spelling performance that take into account possible changes in EEG data statistics after calibration provide for more flexibility in predicting performance. A BCI’s performance can be estimated by analysing the evolution of the character cumulative response function, $\Theta_m(t)$, with data collection. Modeling the target character estimation algorithm with a probabilistic model provides a means to account for shifting data statistics, particularly if the model is parametric. The projected accuracy metric developed by Colwell *et al.* (2014) is an example of a method utilising a probabilistic model to predict BCI performance [53], and is described in the next section.

2.2.2 Projected Accuracy for the CMA Static Stopping Algorithm

Colwell *et al.* (2014) proposed an underlying theoretical model to estimate the projected accuracy of a P300 speller user given a certain amount of data collection by modeling the CMA static stopping algorithm (2.1) with the RCP [53]. The BCI correctly selects the target character in the RCP if the following event occurs: $\{\text{mean target row score} > R - 1 \text{ mean non-target row scores}\} \cap \{\text{mean target column score} > C - 1 \text{ mean non-target column scores}\}$. The accuracy is estimated by analysing the relationship of the mean target classifier score to those of the non-target classifier scores after a fixed amount of data collection.

Given a user’s class conditional classifier likelihoods, $p(y_t|H_0)$ and $p(y_t|H_1)$, for non-target and target classifier scores, respectively, and a P300 speller grid of size $R \times C$, a user’s accuracy with the CMA static stopping algorithm can be estimated for a given RCP sequence limit. The method relies on a series of convolutions and numerical integrations of classifier likelihood functions to obtain the probability density functions (pdf) of the mean non-target or target classifier scores. If the classifier likelihoods are Gaussian, an assumption used in many BCI studies, the derivation is simplified as performance can be characterised by the parameters of the likelihoods, (μ_0, σ_0^2) and (μ_1, σ_1^2) , for non-target and target classifier likelihoods, respectively. The parameters of the classifier likelihoods can be obtained from the training data to initially assess a user’s performance level. The parameters can then be varied accordingly to predict performance for different data statistic shifts.

There are some limitations with the projected accuracy metric developed by Colwell *et al.*. Use of the performance prediction method is limited to cases where the P300 speller uses the RCP and the CMA static stopping algorithm. In addition, visualisation of the estimated performance across various parameter values, $\{\mu_0, \sigma_0^2, \mu_1, \sigma_1^2\}$, is difficult due to a 4-D parameter space. Given the increased inter-

est in the development of other stimulus paradigms and the use of dynamic stopping algorithms, a general method to predict performance than can be parameterised preferably in a 1-D space for easy visualisation, is desirable.

In Chapter 3, a new method is proposed to predict BCI performance based on a probabilistic model of the target character estimation algorithm parameterised by the distance between the class conditional classifier likelihoods. In the case of the Bayesian dynamic stopping algorithm, under a normality assumption, this distance can be quantified with a 1-D parameter called the *detectability index* [54, 55].

2.3 Information Optimisation

The BCI system automatically analyses EEG data in order to determine the message that the user intends to communicate. In the P300 speller, this message is encoded for the user via the stimulus flash pattern, where ideally a P300 ERP is elicited only when the target character is flashed. The magnitude of the ERP response in the EEG data is affected by the parameters of the stimulus pattern, such as the group of characters that are flashed together, the interval between target character presentations, etc. [56, 57]. This ability to modulate the ERP response based on the stimulus presentation parameters potentially provides a mechanism by which to improve P300 speller performance [56, 58].

2.3.1 *Physiological Effects*

The stimulus presentation paradigm defines the flashing pattern for the P300 speller, i.e. the flash group elements, flash group order, etc. Stimulus paradigm parameters, such as the order and timing of stimulus events, have been shown to modulate the elicited P300 responses and to affect performance [56, 57]. For example, the target-to-target interval (TTI) affects the ability of a user to elicit a P300 response with every target presentation due to refractory effects. The amount of time between two

target character presentations determines the strength of the P300 response to the second target character presentation [56]. Several studies have shown that the P300 SNR and detection rate increases with longer TTIs [56, 58].

The effect of TTI on classification performance can be illustrated by analysing the distributions of non-target and target classifier scores as a function of the TTI interval, where the separation between the two distributions is an indicator of system use performance (see Chapter 3). Let TTI X denote the interval between two consecutive target presentations given a sequence of stimulus events, with \mathbf{N} denoting non-target and \mathbf{T} denoting target stimulus events. For example, \mathbf{TNNNT} denotes a sequence with a TTI of 4. Figure 2.5 shows the pdfs of target classifier scores grouped by TTI, as well as the aggregate target and non-target classifier score pdfs for a study participant [59]. For shorter TTIs, the TTI-segregated pdf is closer to the non-target pdf. As the TTI increases, the TTI-segregated pdf approaches and becomes similar to the aggregate target pdf.

2.3.2 Stimulus Presentation Paradigms

While some P300 spellers use a single character presentation, e.g. rapid serial visual presentation (RSVP) [60], the majority of spellers present groups of characters in a single stimulus event in order to increase stimulus presentation rate and spelling speed. However, presenting multiple characters together usually increases the likelihood of classification errors as characters in the same flash group will obtain the same classifier updates. Characters with a high degree of temporal correlation with the target character have been shown to be more often confused with the target character [50, 61].

For the conventional RCP, characters in the same row or column as the target are more likely to be selected in error, particularly adjacent characters in the same row and column [50]. There are other limitations associated with using the RCP.

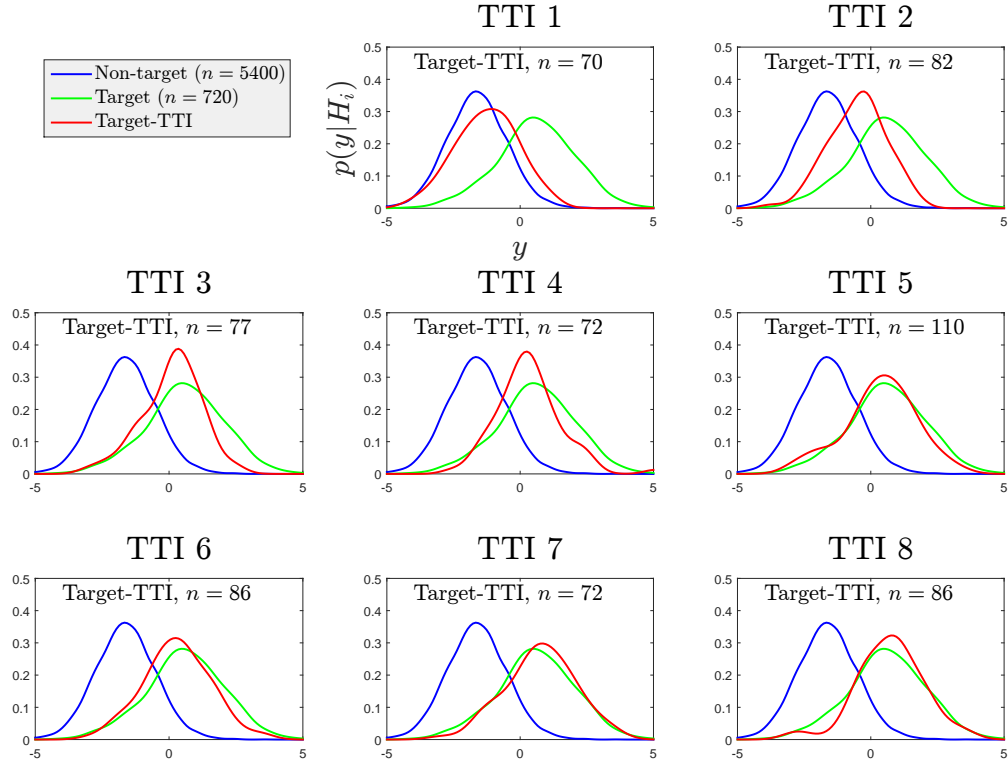


FIGURE 2.5: Effect of target-to-target interval (TTI) on the probability density function (pdf) of the target P300 classifier scores. Classifier scores from a participant were grouped by non-target and target responses. In addition, the target classifier scores were segregated by TTI (Target-TTI). For example, TTI 4 denotes the sequence **TNNNT**, where **T** and **N** are target and non-target stimulus events, respectively. The respective pdfs were obtained via kernel density estimation (KDE) from a histogram of the classifier scores, where n denotes the number of observations used for estimation. In each plot, the blue, green and red curves denote the pdf generated via KDE for the aggregate non-target, aggregate target and the TTI-segregated target classifier scores, respectively.

Due to the randomised order of presentation of row/column flash groups, there is a high likelihood of the occurrence of short TTIs, leading to relatively lower target classifier scores, as illustrated in Figure 2.5. Also, with shorter TTIs, a user might miss a successive target stimulus event, especially if the flash rate is high. Further, anecdotal evidence suggests that users experience visual fatigue with the RCP [50].

Several stimulus paradigms have been designed to address some of the shortcom-

ings of the RCP. Some paradigms have been developed to minimise user distractions, such as the *center speller* that presents groups of characters sequentially at the center of the grid [62], or the *region-based* paradigm that arranges the flash groups in spatially distinct clusters [63]. Other approaches have incorporated elements in the visual interface to increase user focus or elicit additional ERPs that can improve performance, e.g. using colour or familiar faces [64, 65].

Other stimulus paradigms have been designed to reduce the proportion of short TTIs to minimise the negative impact of TTI-related effects. Jin *et al.* (2010) [66] and Polprasert *et al.* (2013) [67] developed paradigms that primarily imposed restrictions on the minimum TTI of potential target characters. Other studies use the checkerboard paradigm (CBP) that was developed by Townsend *et al.* (2010) [50]. The CBP was developed to limit short TTIs and adjacency distraction errors by imposing temporal and spatial constraints on flash group elements generated from virtual matrices of a checkerboard overlay on a speller grid. The CBP was utilised in this work as one of the baseline stimulus presentation paradigms, in addition to the RCP.

In general, most of the proposed changes to stimulus presentation paradigms have focused on cosmetic aspects of the visual paradigm, or the ad-hoc generation and presentation of flash groups. In this work, it is hypothesised that a character’s flash pattern and the character-to-flash group assignment process in a stimulus paradigm can be designed through a more principled process. The role of a BCI is to accurately convey a user’s intent based on the extracting and decoding of relevant information from a user’s brain responses to stimulus events. Consequently, the P300 speller can be approached from an information-theoretic perspective, by exploiting *coding theory* which is the study of the properties of codes and their fitness for a specific application of transmitting information over a noisy channel [23].

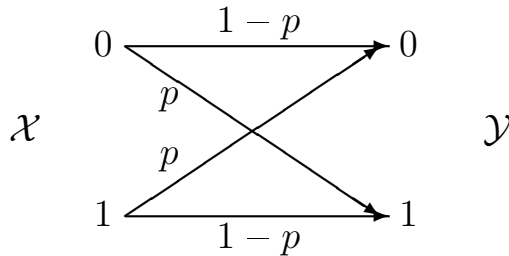


FIGURE 2.6: A binary symmetric channel with input and output alphabets, $\mathcal{X} = \mathcal{Y} = \{0, 1\}$, and a crossover probability during transmission, p .

2.3.3 Communication under Noisy Conditions

The transmission of information correctly from one point to another, usually over a channel, represents a fundamental problem in communication [22, 23]. Information can sometimes be distorted during transmission such that the received message is different from the sent message. In a noisy communication channel, the output, \mathcal{Y} , depends probabilistically on the input, \mathcal{X} . The channel is usually characterised by a conditional probability distribution, $p(y|x)$, which is the probability of receiving y , given that x was sent. For example, the binary symmetric channel (BSC), illustrated in Figure 2.6, is a memoryless channel with binary input and output alphabets, $\mathcal{X} = \mathcal{Y} = \{0, 1\}$, where a transmitted symbol has a probability, p , of being received in error. A channel is denoted as *memoryless* if the output associated with one transmission is independent of the input of previous transmissions, i.e. $p(y_t|x_1, x_2, \dots, x_t) = p(y_t|x_t)$.

With a single channel transmission, it can sometimes be difficult to discern if an error occurred, as is the case with the BSC where the input and output alphabets are identical. While it would be ideal to have noiseless transmission, this usually entails changing the physical characteristics of a system, which can be very expensive or impossible to achieve. An alternative approach is a system solution [22] where the noisy channel is accepted, but redundancies are introduced in the transmission process through repeated or n channel uses, \mathcal{X}^n , via the use of *codewords*. These re-

dundancies are exploited in the decoding process to deduce from the received output sequence, \mathcal{Y}^n , both the sent message and the noise introduced by the channel.

A communication system, illustrated in Figure 2.7, consists of [23]:

- An index set, $[1, 2, \dots, M]$, which represents the number of messages, $\{w_m\}_{m=1}^M$, that can be communicated via the system.
- An encoding function, $\mathbf{X}^n : [1, 2, \dots, M] \rightarrow \mathcal{X}^n$, which maps a message index to an n -length codeword, $\mathbf{x}^n(1), \mathbf{x}^n(2), \dots, \mathbf{x}^n(M)$. Only codewords are transmitted through the channel. A codebook, \mathcal{C} , with a set of n -length codewords and of size, $|\mathcal{C}| = M$ is called an (n, M) -code.
- A decoding function:

$$g : \mathcal{Y}^n \rightarrow [1, 2, \dots, M] \quad (2.2)$$

which is a decision function that determines the sent message, by mapping the received output sequence, \mathbf{Y}^n , to the most likely sent codeword and consequently, the sent message. It can be based on a probability evaluation, $\hat{w} = \underset{m}{\operatorname{argmax}} P(\mathbf{y}^n \text{ received} | \mathbf{x}^n(m) \text{ sent})$, where \hat{w} is the estimate of the sent message.

Due to the introduction of noise during transmission, multiple input sequences can generate the same output sequence, leading to confusable inputs in the decoding process. To minimise the amount of errors in the decoding process, codewords can be selected such that their resulting noisy output sequences are far enough apart to be distinguishable from each other. For the BSC, one approach is to use codewords that differ at many bit positions so that even with a certain number of bit flips, the codeword that is most similar to the received output sequence is that of the sent message.

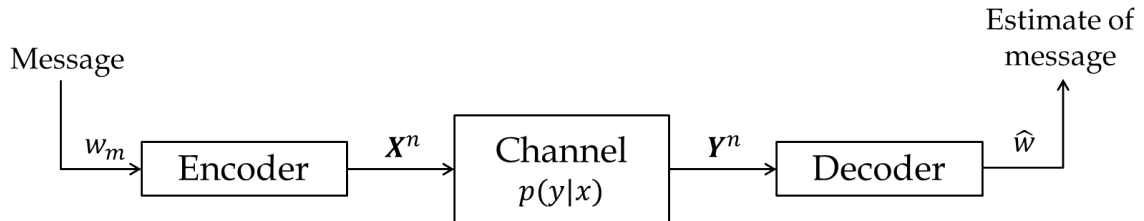


FIGURE 2.7: Schematic of a communication system. A potential message, w_m , is transmitted from one point to another via a communication channel. The channel is characterised by a conditional probability distribution, $p(y|x)$, which is the probability of receiving y given x was sent. The message, w_m , is encoded with a codeword, $\mathbf{X}^n = [x_1, x_2, \dots, x_n]$, representing n channel uses. The transmission of a codeword results in an output sequence, $\mathbf{Y}^n = [y_1, y_2, \dots, y_n]$, at the receiver. The receiver guesses the sent message based on the received output sequence using a decoding rule, $\hat{w} = g(\mathbf{Y}^n)$.

One way to quantify the similarity between codewords is with a metric called the *Hamming* distance [68]. The Hamming distance, d^H , between two codewords is the number of positions where they differ:

$$d^H(\mathbf{x}, \mathbf{y}) = \text{number of } \{i | x_i \neq y_i\} \quad (2.3)$$

For a codebook, \mathcal{C} , the minimum Hamming distance, d_{\min}^H , is the smallest Hamming distance between codewords:

$$d_{\min}^H(\mathcal{C}) = \min(d^H(\mathbf{x}, \mathbf{y}) | \mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}) \quad (2.4)$$

d_{\min}^H affects the number of errors that can occur in the transmission process and still result in an output sequence that is most similar to the sent codeword. The number of transmission errors in a codeword that can be corrected in the decoding process is [69]:

$$\lfloor (d_{\min}^H - 1)/2 \rfloor \quad (2.5)$$

(2.5) is known as the error-correcting capability of a code.

2.3.4 The P300 Speller Communication Channel

A BCI is a noisy communication channel consisting of three units: encoding, modulation and decoding [70]. For the P300 speller, the encoding process dictates the presentation order of all the character choices. A stimulus presentation paradigm determines what characters are flashed in a stimulus event. Each character is a potential message which is encoded for transmission by its flash pattern. The base flash pattern of a stimulus paradigm is its codebook. A codeword represents a character’s flash pattern where each bit represents whether a character is flashed in a stimulus event, e.g. “0” if a character is not flashed and “1” if it is flashed. A sequence represents one instantiation of the codebook with one property of the codebook randomised according to the stimulus paradigm randomisation rules: the character-to-flash group assignments or the order of presentation of the flash groups.

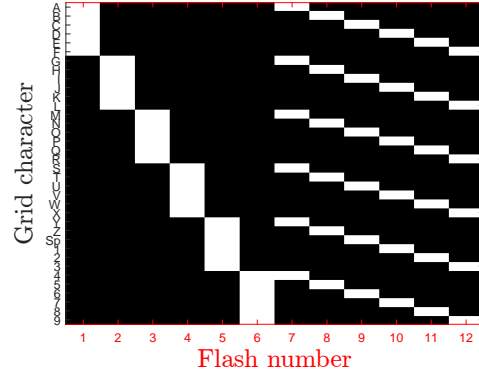
For example, the RCP for an $R \times C$ grid is a $(R \times C, R + C)$ -code. For the 6×6 speller grid in Figure 2.8(a), Figure 2.8(b) shows the codebook for the corresponding RCP. Since the characters in the flash groups are fixed for the RCP, the codeword for each character is unique. After each sequence, the order of presentation of the row/column flash groups is randomised without replacement. However, for some stimulus paradigms, there is character-to-codeword randomisation after each sequence e.g. [50, 71]. Figure 2.8(c) shows the pairwise Hamming distances between codewords for the 6×6 RCP. For one RCP sequence, d^H is either 2 (for characters in the same row or column) or 4, hence $d_{\min}^H(\text{RCP}) = 2$. Repetition of the codebook, i.e. via additional sequences, increases d_{\min}^H . The minimum Hamming distance of a new RCP codebook based on repetitions of a base codebook is:

$$d_{\min}^H = s \times d_{\min}^{H*} \quad (2.6)$$

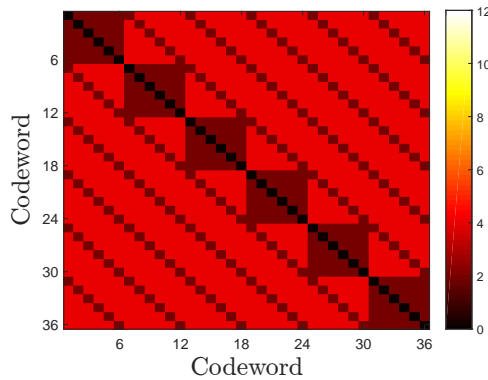
where d_{\min}^{H*} is the minimum Hamming distance of a base codebook; s is the number of repetitions; and d_{\min}^H is the minimum Hamming distance of the new codebook

WORDS(W)					
A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z	Sp	1	2	3
4	5	6	7	8	9

(a) A 6×6 speller grid



(b) Codebook for the 6×6 RCP



(c) Pairwise Hamming distance for the 6×6 RCP

FIGURE 2.8: Encoding process for the row-column paradigm (RCP). (a) The P300 speller grid for a 36-character layout using a 6×6 matrix. (b) Codebook for the RCP with a 6×6 grid. Each row represents the codeword for a character. Each column represents a flash group, with characters highlighted in white when flashed. A sequence represents one instantiation of the codebook with one property of the codebook randomised. A red-coloured axis indicates which codebook property is randomised; for the RCP, the order of presentation of the row/column flash groups. (c) Pairwise Hamming distance between codewords of the 6×6 RCP. Element (i, j) in the grid denotes the Hamming distance, $d^H(\mathbf{c}_i, \mathbf{c}_j)$, between codewords, \mathbf{c}_i and \mathbf{c}_j .

obtained from s repetitions of the base codebook.

A sequence of flash groups, as determined by the stimulus paradigm encoding process, is now presented to the user. Each stimulus event presentation elicits different brain response depending on whether it is a non-target or target stimulus event. The modulation process determines how these stimulus events elicit brain responses

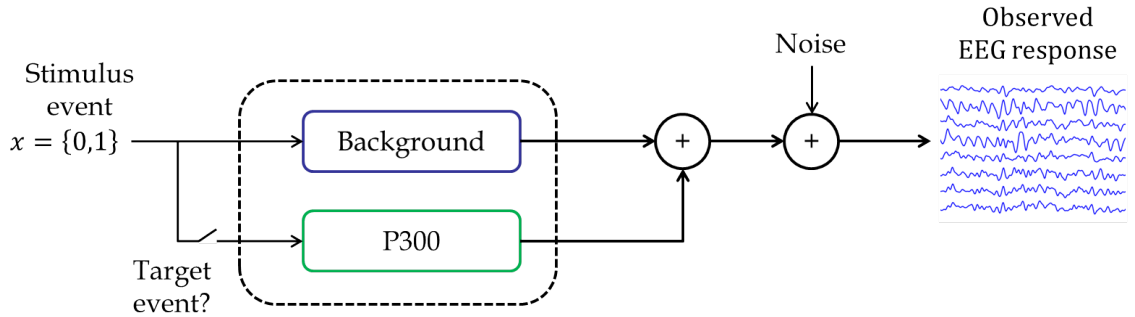


FIGURE 2.9: Illustration of the modulation process in an oddball recognition task. The input to the P300 channel is binary, $x \in \{0, 1\}$, indicating the absence ($x = 0$) or presence ($x = 1$) of an oddball or target stimulus event. A user is presented with a sequence of stimulus events, which all elicit a *background* signal. The random occurrence of a rare target event within the sequence elicits an event-related potential called the P300 signal. These responses are embedded in noise, observed in the electroencephalography (EEG) data.

that are embedded within noisy EEG data. Figure 2.9 illustrates the modulation process involved in the oddball recognition task. Observing a visual stimulus usually elicits a visually evoked response, denoted as a *background* signal. In an oddball paradigm, ideally only flash groups with the target character elicit P300 signals. Thus a non-target flash group elicits only a background signal, while a target flash group elicits combined background and P300 signals.

The decoding process involves signal processing and use of the character cumulative response function to determine which character the user intended to spell. From each time window of EEG data associated with an illuminated flash group, \mathcal{F}_t , a classifier score, y_t , is generated after signal processing. The probability distribution of the classifier score depends on whether the current flash group contains the target character. In effect, a binary input, $x \in \{0, 1\}$, indicating the absence or presence of the target character in a flash group, is mapped to a classifier score random variable,

y , with class conditional likelihoods, $p(y|H_0)$ and $p(y|H_1)$, respectively:

$$x = 0 \rightarrow y \sim p(y|H_0) \tag{2.7a}$$

$$x = 1 \rightarrow y \sim p(y|H_1) \tag{2.7b}$$

The classifier score is used to update the character cumulative response vector, $\{\Theta_m(t)\}_{m=1}^M$, to determine the most likely target character. The performance of the decoding algorithm depends on the ability to determine if a target character is or is not in a flash group, i.e. if a 0 or 1 was sent; specifically the discriminability between the class conditional classifier likelihoods. Errors can be introduced at the modulation stage (e.g. missing a target stimulus event, adjacency distraction errors) or in the decoding stage (e.g. low P300 SNR).

Some approaches in the literature have developed stimulus paradigms based on error-correcting codes [70, 71, 72]. However, these approaches failed to consider physiological effects of stimulus parameters on P300 elicitation, resulting in similar or poorer online performances when compared to the standard RCP. An information-theoretic approach to stimulus paradigm design favours maximising d_{\min}^H since on average, more bits have to be flipped to convert one codeword to another. Achieving higher d_{\min}^H usually results in codewords with a higher proportion of short TTIs, so potential target characters are likely to be flashed more frequently, often in immediate succession, e.g. [71]. However, shorter TTIs especially TTI 1, result in target classifier scores that are closer to the non-target pdf (Figure 2.5). A potential solution was proposed in [70] to account for short TTIs in the decoding process by using a graphical model which accounted for classifier score TTI dependencies. However, this was not enough to compensate for the negative TTI-related effects caused by continuous character presentation.

The development of a new codebook to improve P300 speller performance has to consider the competing interests of information theoretics and the physiological lim-

itations of P300 elicitation. In Chapter 4, a new method is proposed which considers both competing factors in the codebook development process. The proposed method uses objective performance functions in an iterative search of codewords with properties that not only minimise TTI-related effects, but also facilitate the distinction between character choices via their cumulative responses .

2.4 Target Character Estimation

The amount of user data as well as how it is processed and used for character selection plays an important role in enabling the BCI to correctly discern the user’s intended target character. The significant amounts of data required to achieve acceptable accuracy levels contribute to the relatively slow speeds of BCI systems when compared to other commercially available communication alternatives. There is the potential to improve BCI performance through a more informed data collection process, such as one based on a continuous evaluation of the quality of a user EEG responses or the potential message a user intends to communicate.

2.4.1 *Dynamic Data Collection*

The main limitation with static data collection is that it assumes a constant user performance level or similar performance levels if the same amount of data collection is used for all users. However, performance level has been shown to vary across users [73], as well as within users due to the inherent variation in a user’s brain responses during BCI use, or other factors such as attention level and mood [74, 75]. Thus, static data collection is unlikely to be an efficient method of maximising accuracy while minimising spelling time.

An adaptive data collection strategy, termed *dynamic stopping*, can improve performance by giving the BCI system the flexibility to collect more or less data based on continuous evaluation of the quality and strength of a user’s brain responses.

The amount of data collected prior to character selection is controlled by a threshold function. Some approaches rely on summation functions of the P300 classifier scores [48, 49, 76, 77]. Other approaches use a probabilistic model based on the P300 classifier scores, by maintaining a probability distribution over the character choices, updating the distribution following EEG data processing and stopping data collection when a specified confidence level is achieved [60, 66, 78, 79, 80, 81]. In general, dynamic stopping algorithms have been shown to improve user performance when compared to conventional static data collection approaches [24, 82].

Some dynamic stopping algorithms rely on either tailoring the stopping criterion to a user's past performance (which may not be accurate longitudinally) or basing the stopping criterion on data collected from a pool of users. Relying on a pool of users to set the stopping criterion creates the potential for mismatch between the pool and new users. Most dynamic stopping algorithms have not been evaluated in a target user population, such as those with ALS, even though performance improvements or user experiences in studies with healthy participants may not always lead to similar results in disabled participants. Specifically, setting stopping criterion parameters for the target BCI population based on data from healthy participants is likely suboptimal.

2.4.2 Statistical Language Models

The predictability of language is used in everyday text-entry applications to enhance user experience and increase system throughput, e.g. predictive text, auto-correct, machine translation etc. Since the P300 speller is used for communication, language information can be exploited in the target character estimation process by identifying potential likely characters based on a user's spelling history. Language information is usually quantified via a statistical language model which can have varying degrees of complexity, such as based on letters, words, phrases or sentences [83].

The n -gram Model

The collocation of letters in words or words in sentences can be modeled probabilistically via statistical language models [83]. A probability value is assigned to a string of tokens, $\mathbf{w}_1^L = (w_1, w_2, \dots, w_L)$, where tokens can be letters, words, etc. These probabilities are based on a language model developed from a body of text or corpus of a given language.

Given a string of tokens, \mathbf{w}_1^L , the probability assigned to the string via an n -gram model is:

$$P(\mathbf{w}_1^L) = \prod_{i=1}^L P(w_i | \mathbf{w}_1^{i-1}) \quad (2.8a)$$

$$\approx \prod_{i=1}^L P(w_i | \mathbf{w}_{i-(n-1)}^{i-1}) \quad (2.8b)$$

where $P(\mathbf{w}_1^L)$ denotes the probability of a sequence of tokens; $\prod_{i=1}^L P(w_i | \mathbf{w}_1^{i-1})$ denotes the chain rule probability decomposition; and $\prod_{i=1}^L P(w_i | \mathbf{w}_{i-(n-1)}^{i-1})$ denotes the model approximation that reflects the Markov assumption that only the previous $n - 1$ tokens are relevant in determining the conditional probability of the next token [83]. The conditional probabilities, $P(w_i | \mathbf{w}_{i-(n-1)}^{i-1})$, are usually determined by maximum likelihood estimation using a relative frequency count from a corpus.

Any n -gram language model can thus be estimated from a given body of text. However, the number of parameters to estimate for the model becomes exponentially large, M^n , as the order increases, where M is the number of possible elements in the model. As such, most language-based applications are limited to bigram ($n = 2$) and trigram ($n = 3$) language models. Figure 2.10 shows an example probability distribution for a letter bi-gram language model developed from the Carnegie Mellon University corpus [84]. Each element in the grid, (i_{row}, j_{column}) , denotes the conditional probability, $P(A_j | A_i)$, that the user will spell the j^{th} letter in the alphabet, A_j ,

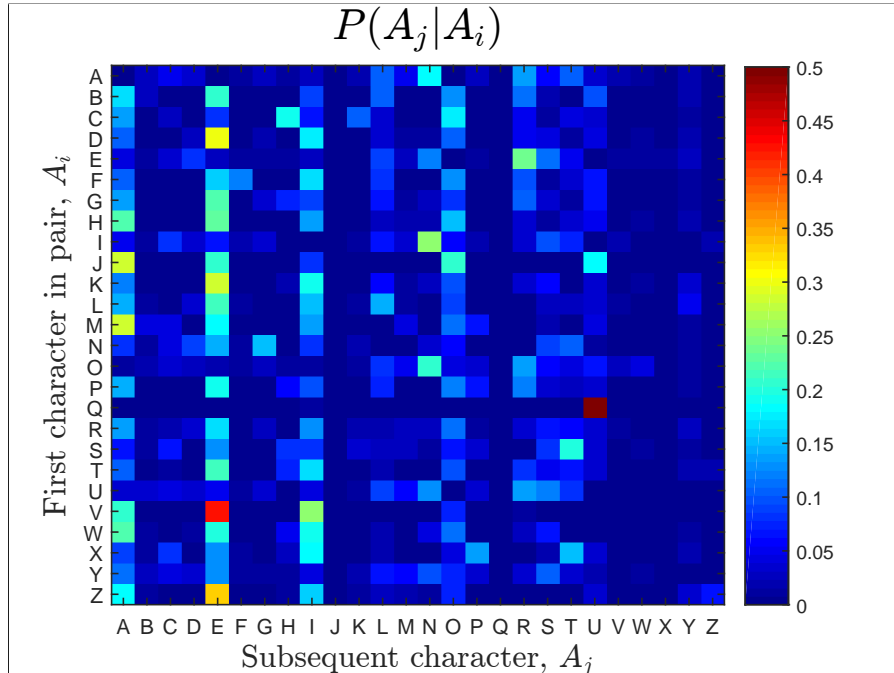


FIGURE 2.10: The character probability matrix for a letter bi-gram language model. The probability matrix was developed from the Carnegie Mellon University dictionary [84]. Element (i_{row}, j_{column}) in the grid denotes the conditional probability, $P(A_j | A_i)$, that the user will spell the j^{th} letter in the alphabet, A_j , given the i^{th} letter, A_i , was previously spelled. Probability values are clipped at 0.5 for visualisation.

given the i^{th} letter, A_i , was previously spelled. For example, vowels are more likely to follow consonants, and specifically, U is most likely to follow Q , compared to E or S . There can be contextual variation in the probability distribution depending on the language, the body of text, etc.

BCI Application

Recent studies focusing on incorporating the predictive capacity of language in ERP-based spellers have shown promise in improving performance [85, 86]. One approach is to incorporate word suggestions into the user interface. Ryan *et al.* (2011) developed a P300 speller used in conjunction with a predictive spelling program [86]. However, the increased task difficulty of selecting numbered word options from a separate drop down menu negatively affected performance as a result of the effect

of increased cognitive load on P300 signal amplitudes. The manner of integration of predictive word options is an important concern, especially for disabled users who might experience additional challenges in navigating a changing interface. Kaufmann *et al.* (2012) developed a P300 speller interface which minimised cognitive load by integrating the predicted word options directly into the speller grid as a new column [87]; hence the word options were selected in a similar manner to conventional character selection.

Other approaches integrate the language model within the character selection algorithm without changing the user interface. Orhan *et al.* (2012) [60] and Speier *et al.* (2014) [81] enhanced their probabilistic dynamic data collection algorithms with language models, which resulted in statistically significant improvements in online performance. However, these approaches need to be evaluated online in target BCI populations.

The next section describes the BCI decision-making algorithm used in this work, a dynamic data collection strategy that incorporates language, which was developed and tested online in non-disabled and target BCI users.

2.5 Bayesian Dynamic Stopping

This work builds on an adaptive data collection algorithm developed by Throckmorton *et al.* (2013), which uses a Bayesian method to control data collection based on a confidence level that a character is the correct target [25]. This dynamic stopping approach allows for the flexibility of controlling data collection, with user-specific controls, based on continuous flash-to-flash assessments of the quality and strength of a user’s EEG responses. For example, the system collects more data under low SNR conditions and less data under high SNR conditions.

Probabilistic data collection algorithms provide a convenient framework to include additional knowledge, such as language, to further inform the BCI target char-

acter estimation process without having to redesign the user interface. The Bayesian dynamic stopping (DS) algorithm can be enhanced with a statistical language model by incorporating prior information about the likelihood of characters based on a user’s spelling history. Preliminary results from online BCI studies non-disabled participants and participants with ALS are also presented which show significant improvements in performance from static data collection and the possibility of further enhancements with a language model.

2.5.1 Algorithm Implementation

The cumulative character response function is a probability distribution, $\{P_m(t)\}_{m=1}^M$, which is maintained over the character choices. This probability distribution represents the confidence level that each character is the target character, given a user’s EEG responses. New information via EEG data collected after each stimulus event is integrated into the probability model via a Bayesian update process. Data collection is stopped when a character probability attains a preset probability threshold level within a data collection limit. After data collection, the character with the maximum probability is selected as the user’s intended target character.

The Bayesian DS algorithm consists of an offline portion and an online portion. A flow chart of the algorithm’s implementation is shown in Figure 2.11. In the offline portion, the likelihood pdfs of the non-target and target classifier scores are estimated by smoothing out the respective histograms of classifier scores from the user’s training data via kernel density estimation (KDE). The estimated likelihood pdf, $p(y|H_0)$ and $p(y|H_1)$, for the target and non-target scores, respectively, are used in the online Bayesian update process.

In the online portion, prior to data collection each character, C_m , is assigned an initial probability of being the T^{th} target character, C_T^* , $P_m(0) = P(C_m = C_T^*|\mathbf{Y}_0, \mathcal{F}_0)$, where \mathbf{Y}_0 denotes the initial empty classifier score vector. The initialisa-

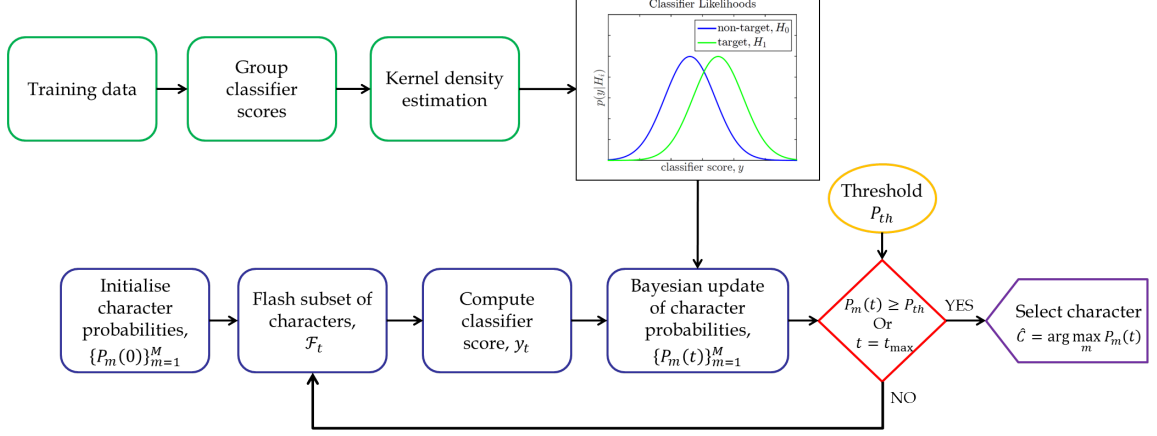


FIGURE 2.11: Bayesian dynamic stopping algorithm for the P300 speller. Prior to data collection, character probabilities are initialised, either from a uniform distribution or a language model. With each new flash, character probabilities are updated with Bayesian inference, using the target and non-target pdfs, (developed from training data via kernel density estimation), until the preset threshold probability is met or the data collection limit is attained. A data collection limit, t_{\max} , is imposed for practical reasons as convergence is not always guaranteed. After data collection is stopped, the character with the maximum probability is selected at the user’s intended target character.

tion probability, $P(C_m = C_T^* | \mathbf{Y}_0, \mathcal{F}_0)$, reflects *a priori* knowledge about the likelihood of the character to be selected as the target. If it is assumed that each character is equally likely to be the target, then the initialisation probability is $\frac{1}{M}$, where M is the number of possible target character choices.

After a subset of characters in the grid, \mathcal{F}_t , is flashed on the screen, the resulting classifier score, y_t , is used to update the character probabilities by Bayesian inference:

$$P(C_m = C_T^* | \mathbf{Y}_t, \mathcal{F}_t) = \frac{P(C_m = C_T^* | \mathbf{Y}_{t-1}, \mathcal{F}_{t-1}) \times p(y_t | C_m = C_T^*, \mathcal{F}_t)}{\sum_{j=1}^M P(C_j = C_T^* | \mathbf{Y}_{t-1}, \mathcal{F}_{t-1}) \times p(y_t | C_j = C_T^*, \mathcal{F}_t)} \quad (2.9)$$

where $P(C_m = C_T^* | \mathbf{Y}_t, \mathcal{F}_t)$ is the posterior probability of character C_m being the T^{th} target character, C_T^* , given the observed classifier score vector, \mathbf{Y}_t , and the current subset of flashed characters, \mathcal{F}_t ; $P(C_m = C_T^* | \mathbf{Y}_{t-1}, \mathcal{F}_{t-1})$ is the prior probability of character C_m being the T^{th} target character; $p(y_t | C_m = C_T^*, \mathcal{F}_t)$ is the likelihood of the current classifier response, y_t , given that the character was or was not present in \mathcal{F}_t ; and the denominator normalises the probabilities over all the character probabilities.

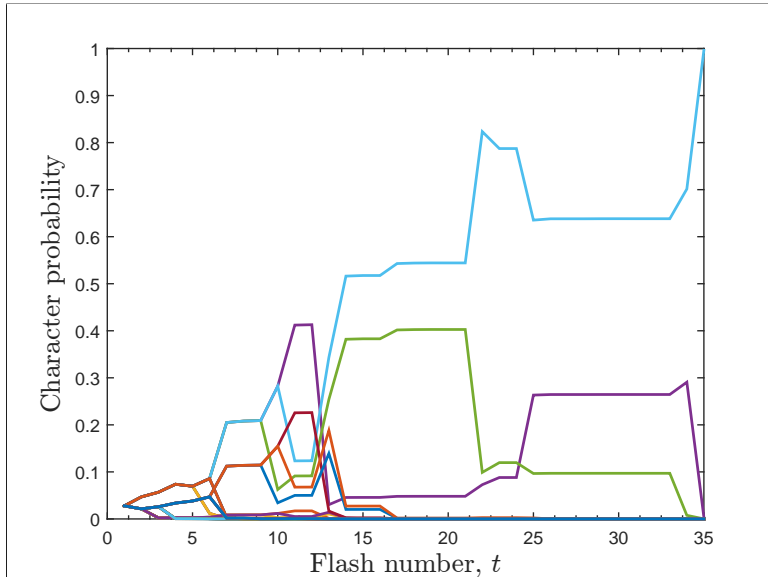


FIGURE 2.12: Evolution of character probabilities in Bayesian dynamic stopping. Prior to data collection, character probabilities are initialised. With each flash, character probabilities are updated via Bayesian inference. After several flashes, there is separation between likely and unlikely characters which increases with more data collection. With more data collection, the probability of one character converges to 1, and ideally this should correspond to the target. Data collection is stopped when a character’s probability attains the preset threshold value (set at 0.9) or the data collection limit is reached. The character with the maximum probability after data collection is selected as the target character.

The character likelihood, $p(y_t|C_m = C_T^*, \mathcal{F}_t)$, is assigned from the likelihood pdfs according to:

$$p(y_t|C_m = C_T^*, \mathcal{F}_t) = \begin{cases} p(y_t|H_0), & C_m \notin \mathcal{F}_t \\ p(y_t|H_1), & C_m \in \mathcal{F}_t \end{cases} \quad (2.10)$$

After each Bayesian probability update, the character probabilities are compared to a threshold, P_{th} , to determine which character is to be selected as the target. Data collection is stopped, at t_s , when a character’s probability attains the threshold value or the data collection limit, t_{max} , is attained:

$$t_s \triangleq \min(t : P_m(t) \geq P_{th}, t_{max}) \quad (2.11)$$

A data collection limit is usually imposed for practical reasons as convergence is not always guaranteed. When data collection is stopped, the character with the

maximum probability is selected as the user's intended target, $\hat{C} = \underset{m}{\operatorname{argmax}} P_m(t_s)$.

Figure 2.12 shows the evolution of character probabilities ($M = 36$) from a simulation of a P300 speller character selection process using EEG data, for a character that was correctly selected. While all characters are initialised with the same probability value, after some amount of data collection, there is separation of likely and unlikely characters. The few likely characters are usually in the same flash group as the target, e.g. in the RCP, in the same row or column as the target character. With even more data collection, the probability of one character starts to converge towards 1, and ideally this should be the target character.

2.5.2 Inclusion of a Language Model

Information about the likelihood of characters can easily be integrated within the framework of probabilistic data collection algorithms without affecting algorithm structure or the BCI task difficulty. The Bayesian DS algorithm offers a built-in mechanism to include a statistical language model based on a user's spelling history at the probability initialisation step prior to the character selection process. The inclusion of a language model in a dynamic stopping algorithm is termed *dynamic stopping with a language model* (DSL_M).

The character probabilities can be viewed as an M -dimensional random vector, $[P_1(t), P_2(t), \dots, P_M(t)]$. With data collection, the vector moves randomly within an M -dimensional space towards sparse probability vectors where the probability mass is concentrated on one character, $\lim_{t \rightarrow \infty} P_{\max}(t) = 1$. It is hypothesised that the inclusion of the language model can potentially reduce character selection time and improve accuracy by biasing the maximum probability towards likely target characters and reducing the number of probability update steps required to attain the threshold value.

Given a string of BCI character selections, $\mathcal{A}_1^{T-1} = a_1, a_2, \dots, a_{T-1}$, these previous selections can provide some predictive information about the next character selection. The n -gram model (Section 2.4.2) is used to set the initialisation probabilities for the subsequent character based on a weighted combination of the language model probabilities and a uniform distribution:

$$P(C_m = C_T^* | \mathbf{Y}_0, \mathcal{F}_0) = \alpha P(C_m | \mathcal{A}_{T-(n-1)}^{T-1}) \left(1 - \sum_{NAC} \frac{1}{M} \right) + (1 - \alpha) \frac{1}{M}, \quad C_m \in \{A : Z\} \quad (2.12a)$$

$$\text{or } \frac{1}{M}, \quad C_m \notin \{A : Z\} \quad (2.12b)$$

where $P(C_m = C_T^* | \mathbf{Y}_0, \mathcal{F}_0)$ is the initialisation probability of character C_m being the T^{th} character in the target word, C_T^* ; $P(C_m | \mathcal{A}_{T-(n-1)}^{T-1})$ is the probability that the next character is C_m given the previously selected characters, $\mathcal{A}_{T-(n-1)}^{T-1}$; α denotes the weight of the language model; $1 - \alpha$ denotes the weight of a uniform distribution, which is an error factor to account for possible misspellings; and $\sum_{NAC} \frac{1}{M}$ is the sum of the non-alphabetic character (NAC) probabilities, which is subtracted from 1 to normalise the probabilities.

As a preliminary step towards investigating the potential for language models to improve spelling accuracy and communication rate, a simple letter bigram model ($n = 2$) was implemented. In Mainsah *et al.* (2014), the Bayesian DS algorithm with a bigram language model was tested in non-disabled participants and resulted in statistically significant improvements in online P300 speller performance [59]. Compared to a DS algorithm with a uniform initialisation process, the inclusion of a language model resulted in statistically significant reduction in character selection times, without a degradation in accuracy levels.

Given these positive results, the enhanced algorithm was then tested online in par-

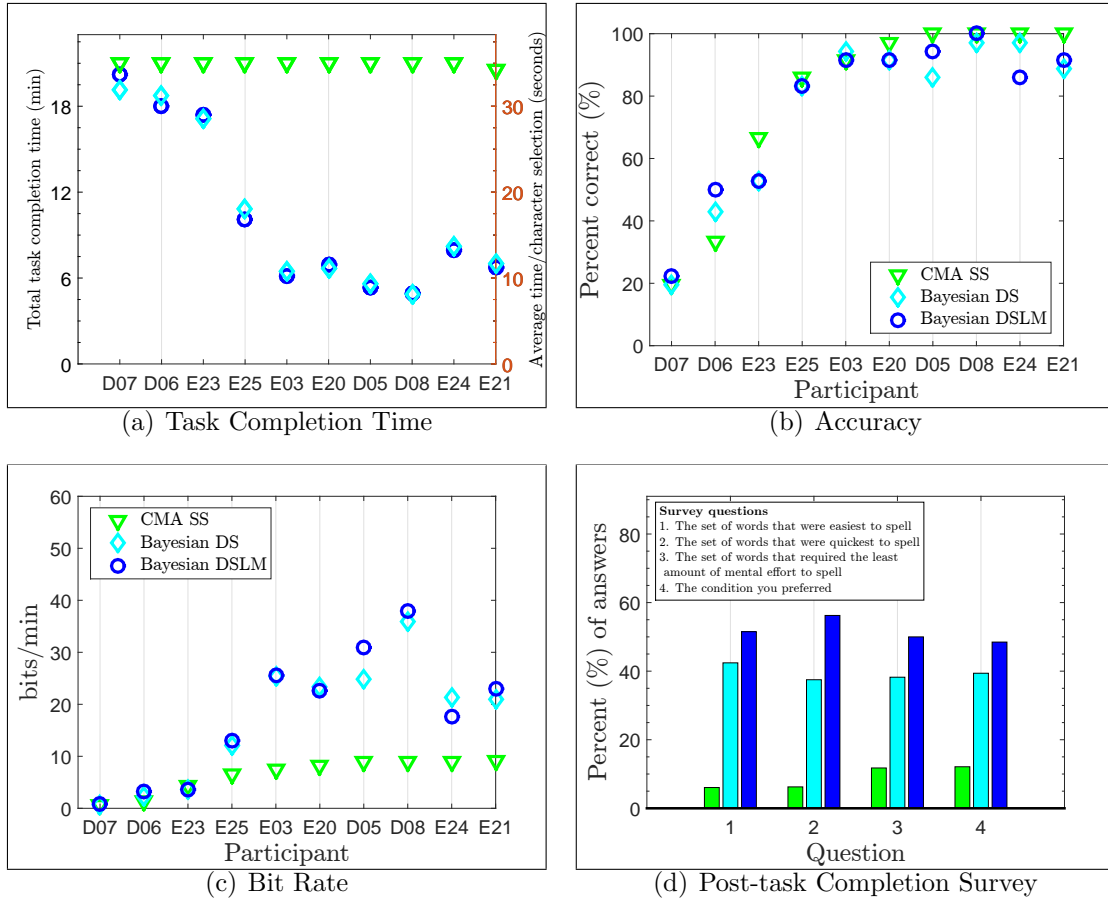


FIGURE 2.13: Comparison of performance measures of ALS participants using different data collection algorithms: cumulative moving average static stopping algorithm (CMA SS), and Bayesian dynamic stopping (DS) and dynamic stopping with language model (DSLML).

participants with ALS to validate the results in a target BCI population. The Bayesian DS and DSLM data collection algorithms were compared against static data collection in participants with ALS who performed single word copy-spelling tasks with the P300 speller. Participants were also polled on their algorithm preference via a series of questions following the spelling task.

Figure 2.13 shows participant performance measures, as well as the participant survey results from the ALS participant study presented in Mainsah *et al.* (2015a) [88]. Figure 2.13(a) shows the average time it took to spell each character, as well

as the total spelling task completion (6 six-lettered words). Compared to static data collection at 35 seconds per character selection, participants observed a statistically significant decrease in character selection time (45-75%), mostly ranging from 8-20 seconds/character selection, $p < 10^{-3}$. Despite the significant reduction in the amount of data collected with the dynamic stopping algorithms, accuracy levels did not significantly deteriorate from static data collection levels, as shown in Figure 2.13(b), $p < 0.23$.

The combination of maintaining similar accuracy levels while statistically significantly reducing data collection resulted in a significant improvement in bit rate from static (6.44 ± 3.21 bits/min) to DS (17.06 ± 11.78 bits/min) and DSLM (17.82 ± 12.54 bits/min), $p < 10^{-5}$, Figure 2.13(c). A statistically significant improvement was also observed for theoretical bit rate (TBR), which excludes time pauses between character selections: static (7.13 ± 3.56 bits/min), DS (25.22 ± 19.56 bits/min) and DSLM (26.71 ± 21.21 bits/min). Since the time pauses between character selections can be varied based on user preference, TBR represents an upper bound of a user's possible communication rate. Based on the post-session survey results, shown in Figure 2.13(d), participants overwhelmingly preferred the two dynamic data collection algorithms.

Unlike in the non-disabled study presented in [59], there was no significant improvements from the dynamic stopping algorithm with uniform character initialisation to that with the inclusion of the letter bigram language model in the study with ALS participants. There is likely more variability in ALS population performance due to other confounding factors such as differences in disease etiology and progression, in contrast to the non-disabled population which tends to be more uniform and skewed towards a younger demographic. Using a letter bigram language model might be overly simplistic and a more complex language model that takes into account all of the user's spelling history, such as trigram or higher order language models, could

potentially improve performance. The spelling task designed in this study, like most P300-based BCI studies, involved single word copy-spelling tasks where word length is known a priori. However, a more realistic BCI system would test the algorithm on phrases or sentences. The complexity of the language model can be expanded to include words or phrases within a sentence structure.

In Chapter 5, a new target character estimation algorithm is proposed by increasing the complexity of the language model used in the Bayesian DS algorithm to facilitate BCI use for phrase or sentence spelling tasks. A higher order language model based on all of the user's spelling history is used in the probability initialisation step, in a manner that minimises the need for erroneous character revisions.

2.6 BCI Error Correction

It is unrealistic to expect a BCI system to perform with perfect accuracy as the BCI decision process is error prone since it must decode information from noisy data. BCI decision errors can be handled in two ways: by having the user actively attempt to correct the error, or by having the BCI system attempt to detect errors and correct them. However, it should be noted that ERP-based BCI spellers are already limited by their slow character selection times and a corrective mechanism can further increase the spelling time. The benefit of an error-corrective mechanism should be evaluated against any potential increase in spelling time.

2.6.1 Active User Correction

Active user correction is the most common method for error correction. A *backspace* or *delete* function is included as one of the BCI character choices, which the user can select to correct an erroneous BCI selection. However, there is the possibility of making errors in the correction process. This limits BCI system use to only users with high accuracy levels. Taking into account erroneous character revisions, it has

been recommended that ERP-based BCI speller systems perform with accuracies $> 70\%$ [12], to be used as effective communication systems. Alternatively, automatic error detection and correction strategies by the BCI can save time in making active erroneous character revisions by the user and thus improve system efficiency.

2.6.2 *ErrP-based Correction*

One approach for automatic error detection that has been proposed in the literature is to take advantage of EEG-based responses elicited as a result of errors [89]. Error-related potentials (ErrP) are ERPs elicited when a person is aware of or perceives erroneous behaviour. ErrPs have been suggested as inputs to drive corrective mechanisms by detecting their occurrence following BCI presentation of the selected character [90, 26]. The detection of ErrPs could be used to veto erroneous actions by the BCI.

A few studies have implemented ErrP-based corrective mechanisms online in P300 spellers following BCI character selection, with automatic character deletion if an ErrP is detected [27, 28, 91, 92]; this mechanism will be denoted as a P300-ErrP cascade in this work. Perrin *et al.*, (2012) [91] implemented an additional corrective action following ErrP-based deletion, by substituting the deleted character with the next most probable character. Spüler *et al.*, (2012) [92] proposed and tested their ErrP-based correction system in non-disabled users and people with ALS, with the information transfer rates of participants in both groups improving with ErrP-based correction.

Unlike with the P300 speller where character selection is made after a certain amount of data collection, ErrP-based correction is based on single trial ERP detection following presentation of the BCI selected character. The performance of the ErrP classifier depends on the underlying pdfs of the ErrP classifier scores of correct and erroneous character selections, which is affected by the amount of data

used to train the classifier. A common complaint with training ErrP classifiers for P300 spellers is the long time required to collect enough ErrP classifier training data. For example, using the Townsend *et al.* (2010) 9×8 speller grid with the checkerboard paradigm [50] at 24 flashes/sequence, a typical amount of training data of 5 sequences/character would result in 120 labeled samples/character with which to train the P300 classifier. However, presenting a selected character as feedback for user evaluation will result in 1 labeled sample/character to train the ErrP classifier.

In general, online performance of ErrP-based correction mechanisms have been mixed, with detection of ErrP classifiers ranging from 60 – 90% accuracy, 40 – 60% sensitivity (hit rate) and 80 – 90% specificity. Another limitation with ErrP-based correction is the additional computational resources required with the incorporation of an ErrP-based signal processing unit to the BCI speller framework. Since the P300 speller is used for linguistic communication, language-based error detection and correction can provide a less expensive alternative to ErrP-based corrective mechanisms.

2.6.3 Language-based Correction

Language-based correction, such as spell-check, auto-correction, etc., is an ubiquitous feature in most text-entry systems or communication devices. Spelling correction algorithms that exploit various natural language processing strategies have been well-studied for a wide range of applications. The most common method to detect spelling errors is to check if a word is found in a dictionary and declare an error if the word is not present. Error correction can then be achieved by looking at words that closely match the out-of-vocabulary (OOV) word; in general the bigger the vocabulary size, the better the correction performance. While an exhaustive search of the whole dictionary can be done, a narrower list of candidate words, $\mathcal{D} = \{D^1, D^2, \dots, D^J\}$, is generally used by exploiting the positional context of errors

to limit the word choices to those that are within a similarity range from the OOV word. For example, the Levenshtein edit distance is the minimum number of single character edits, insertions, deletions and substitutions needed to convert from one string to another [93].

However, it is also necessary to select the most likely target word from a list of candidate choices. One method is word frequency in a given body of text, which can be quantified via a statistical language model, where $P(D^j)$ represents the probability of occurrence of a word, D^j . However, additional contextual information about the likelihood of characters in the word choices can provide a better estimate of the target word. This can be characterised via an error model that quantifies the likelihood that one of the word alternatives is the intended target choice given an input word [94].

The *noisy channel* model can be used to estimate the target word by combining the language and error models to do spelling correction [95]:

$$P(D^j|W) \propto P(W|D^j)P(D^j) \quad (2.13)$$

$$\hat{C} = \operatorname{argmax}_{D^j \in \mathcal{D}} P(W|D^j)P(D^j) \quad (2.14)$$

where $P(D^j|W)$ is the posterior probability of the word D^j , given the spelled word, W ; $P(W|D^j)$ is the likelihood of the spelled word given the candidate word D^j (the error model); $P(D^j)$ is the prior probability of D^j (the language model); and \hat{C} is the corrected word choice.

A noisy channel model for spelling correction can be used in any application where a probabilistic error model can be quantified, e.g. optical character recognition, speech recognition, machine translation, etc. However, it has not been applied to P300 spellers for error correction. In the P300 speller context, an error model can be derived from the character cumulative responses, $\{\Theta_m(t)\}_{m=1}^M$, as they provide in-

formation on the likelihood of character choices given all of a user’s EEG responses. Unlike the time-intensive system for developing an ErrP classifier, the character cumulative responses are already available from the target character estimation process at no additional cost. $\{\Theta_m(t)\}_{m=1}^M$ can conveniently be used within a noisy channel model for error detection and spelling correction.

In Chapter 6, the relative benefits of different error-correction mechanisms for BCI applications will be analysed and compared: active user correction, ErrP-based correction and language-based correction. In the chapter, a new language-based error-correction algorithm is proposed for the P300 speller that potentially achieves a balance in increasing accuracy with minimal increase in spelling time, by using a noisy channel model with an error model that exploits information already available from the P300 speller character selection process.

2.7 Summary

The operation of the P300 speller consists of three main parts: the visual interface which controls how ERPs are elicited in the brain; the amount of data collection and BCI decision-making algorithm which determine the accuracy and speed of the system; and the selection and presentation of intended user’s choice. It is hypothesised that each of these areas have the potential for improvement. The Bayesian dynamic stopping algorithm provides a framework for further enhancements in all three areas.

The rest of this document is organised as follows. Chapter 3 describes a new method to predict P300 speller performance using P300 classifier discriminability based on a probabilistic model of a BCI decision-making algorithm; and presents results to verify and validate the proposed performance prediction method using the Bayesian dynamic stopping algorithm. Chapter 4 describes a new method which considers the competing interests of information theoretics and physiological limitations of P300 elicitation to design an improved stimulus paradigm; and presents results

comparing the new stimulus paradigm with the conventional row-column paradigm. Chapter 5 describes a new target character estimation algorithm, a Bayesian dynamic stopping algorithm with a more complex language model that minimises the need for erroneous character revisions; and presents results comparing the new algorithm with the Bayesian dynamic stopping with a simpler language model. Chapter 6 presents results comparing various error-correction methods; and includes a new language-based error correction method for P300 spellers. Chapter 7 summarises the contributions of the previously mentioned studies.

Predicting BCI Performance with the Classifier Detectability Index

Several methods in the literature have been proposed to predict BCI performance. However, most of these methods either rely on estimates derived from a user's training data, which may not be a good indicator of future performance, or estimates of performance derived from a pool of users, making the predictions susceptible to user variation. Developing and analysing a theoretical model of a generic BCI user potentially addresses the issues of not relying on empirical estimates either from a user or a pool of users, and accounts for the possibility of shifts in performance levels after calibration.

Colwell *et al.*, (2010) proposed an underlying theoretical model to estimate the projected accuracy for a user given a certain amount of data collection based on a probabilistic model of the P300 speller [53]. However, the Colwell *et al.* method was limited to only considering the cumulative moving average (CMA) static stopping algorithm (2.1) and the row-column stimulus presentation paradigm. Given the increased interest in dynamic stopping algorithms and alternative stimulus presen-

tation paradigms, a generalised approach to performance prediction is desirable.

A user’s P300 speller performance depends on how well the classifier can distinguish between target and non-target EEG responses. The classifier’s ability to discriminate between the two types of responses can be quantified by a metric called the *detectability index* [55]. A new method is proposed to estimate BCI performance for the Bayesian dynamic stopping algorithm [25] from the detectability index either analytically or empirically via Monte Carlo simulations. The proposed performance prediction method is applicable not only to the row-column stimulus presentation paradigm, but also to other stimulus presentation paradigms. The method relies on data that is collected during system calibration and accounts for possible changes in user performance after calibration. The new method is verified with offline analyses using simulations with synthetic EEG data and validated with experimental results from online BCI studies.

3.1 Theoretical Model

P300 speller performance can be determined by analysing the evolution of the character cumulative response, $\Theta_m(t)$, a random variable, at each time index, t , as data collection proceeds. $\Theta_m(t)$ determines how likely a character, C_m , is the target character, with data collection. $\Theta_m(t)$ depends on the cumulative classifier responses and a character’s flash pattern. The BCI correctly selects the target character, C^* , if at the time data collection is stopped, t_s , the target character’s cumulative response exceeds the maximum response of all of the non-target characters, i.e. $\hat{C} = C^*$ if $\Theta_i(t_s) > \max_{j \neq i} \Theta_j(t_s) | C_i = C^*$. The overall performance depends on how well the BCI system can distinguish between target and non-target EEG responses. For example, in the case of the Bayesian DS algorithm, performance depends on the discriminability of the class conditional likelihoods, $p(y|H_0)$ and $p(y|H_1)$.

Given $C_i = C^*$, let $Y = \Theta_i(t_s)$ and $X_{\max} = \max_{j \neq i} \Theta_j(t_s)$. Determining the accuracy of character selection requires evaluating the average probability of the random inequality, $P(X_{\max} < Y)$ (Appendix B), after determining the probability distribution of the maximum of a finite number random variables, $X_{\max} = \max(X_1, X_2, \dots, X_N)$ (Appendix C). This approach was used in Colwell *et al.*, (2010) to determine the projected accuracy for the CMA static stopping algorithm (2.1) using a row-column paradigm (RCP). The amount of data collection prior to character selection can be fixed or dynamically determined. With a dynamic stopping algorithm, data collection is stopped when any character's cumulative response attains a pre-set threshold value or the data collection limit is attained. The average amount of data collection prior to character selection depends on the rate of convergence of $\Theta_m(t)$ to the stopping threshold, within the data collection limit.

3.1.1 The Bayesian Probability Random Variable

For the Bayesian DS algorithm, the random variables that are analysed to evaluate system performance are the Bayesian probabilities, $\{P_m(t)\}_{m=1}^M$. The system correctly selects the target character if the maximum probability corresponds to that of the target character, $\hat{C} = C^*$ if $P_i(t_s) > \max_{j \neq i} P_j(t_s) | C_i = C^*$. For simplicity, the character likelihood in the probability update step, $p(y_t | C_m = C^*, \mathcal{F}_t)$ (2.10) will be denoted as $p_m(t)$. $p_m(t)$, which is incorporated into the model at each probability update step is also a random variable and can be assigned $p(y_t | H_0)$ or $p(y_t | H_1)$ depending on whether character C_m is present or absent in a flash group, \mathcal{F}_t .

For computational simplicity, instead of (2.9), the Bayesian probabilities can be expressed in a logarithmic function:

$$\ln P_m(t) = \ln P_m(t-1) + \ln p_m(t) - \ln \left(\sum_{j=1}^M p_j(t) P_j(t-1) \right) \quad (3.1)$$

where $\ln P_m(t)$ is a random sum (Appendix A). Let S be a sum of T random variables, such that $S(T) = Z(1) + Z(2) + \dots + Z(T)$, and T is also a random variable. In the context of the Bayesian probabilities, $Z_m(t) = \ln p_m(t) - \ln \left(\sum_{j=1}^M p_j(t) P_j(t-1) \right)$ and $S_m(t) = P_m(t)$. The latter term in $Z_m(t)$ makes the random variable analysis more complex. The uncertainty in the likelihood assignments (2.10) needs to be integrated out of the model given the presentation of random flash groups during character selection. Nonetheless, as a consequence of the binary choice in the likelihood function, the ratio, $p(y|H_1)/p(y|H_0)$, commonly referred to as the *likelihood ratio*, can instead be considered.

The *likelihood ratio test* is a well-established statistical analysis tool used during decision making in binary hypothesis testing [44]. The likelihood ratio test has an equivalent formulation in the Bayesian domain. By operating in the likelihood ratio domain, a user's performance level can be parameterised with a single value via a measure called the *detectability index* [55] which quantifies the discriminability between two normal distributions; in this case that of the target and non-target classifier scores. To simplify the random variable analysis, this work proposes and demonstrates a theoretical model where modeling P300 speller operation in the likelihood ratio domain is equivalent to modeling in the Bayesian probability domain. Following this substitution with the likelihood ratio domain, the performance measures for the Bayesian DS algorithm are derived.

3.1.2 The Likelihood Ratio Test

Statistical hypothesis testing involves the process of determining how likely it is that a set of observations are to be generated under a given statistical hypothesis. Each hypothesis represents a probabilistic model of a random variable, $H_i : \mathbf{x} \sim p(\mathbf{x}|H_i)$, where $p(\mathbf{x}|H_i)$ is the likelihood of observation \mathbf{x} under the hypothesis, H_i . The likelihood ratio test (LRT) is most commonly used for binary hypothesis testing [44].

A decision is made to accept or reject a hypothesis after comparing the likelihood ratio, $\lambda(\mathbf{x})$, to a threshold value, η :

$$\lambda(\mathbf{x}) = \frac{p(\mathbf{x}|H_1)}{p(\mathbf{x}|H_0)} \quad (3.2)$$

where the *null*, or H_0 hypothesis is selected if $\lambda(\mathbf{x}) < \eta$ or the *alternative* or H_1 hypothesis is selected if $\lambda(\mathbf{x}) \geq \eta$. If the hypotheses have prior probabilities, $P(H_0)$ and $P(H_1)$, respectively, the threshold value that minimises the probability of error, p_e , given a 0-1 loss function is $\eta = P(H_0)/P(H_1)$ [44].

In general, for a multihypothesis or an M -ary hypothesis test, it can be determined probabilistically how likely it is that a set of observations were generated under a given hypothesis by using Bayes' rule. Another minimum p_e detector can be considered which minimises the Bayes' risk. From Bayes' rule, the posterior probability of a given hypothesis given an observation, $P(H_i|\mathbf{x})$, can be determined by incorporating the likelihood of the observation and the hypothesis prior:

$$P(H_i|\mathbf{x}) = \frac{p(\mathbf{x}|H_i)P(H_i)}{p(\mathbf{x})} \quad (3.3)$$

The hypothesis with the maximum *a posteriori* probability (MAP) is selected as the true hypothesis. The Bayesian DS algorithm is an example of a MAP decision test.

When $M = 2$, i.e. in a binary hypothesis test case, by rearranging (3.3), the

Bayesian probability can be expressed as a function of the likelihood ratio:

$$\begin{aligned}
 P(H_1|\mathbf{x}) &= \frac{\frac{p(\mathbf{x}|H_1)P(H_1)}{p(\mathbf{x}|H_0)P(H_1)}}{\frac{p(\mathbf{x}|H_1)P(H_1)}{p(\mathbf{x}|H_0)P(H_1)} + \frac{p(\mathbf{x}|H_0)P(H_0)}{p(\mathbf{x}|H_0)P(H_1)}} \\
 &= \frac{\lambda(\mathbf{x})}{\lambda(\mathbf{x}) + \frac{P(H_0)}{P(H_1)}} \tag{3.4a}
 \end{aligned}$$

$$P(H_0|\mathbf{x}) = \frac{\lambda(\mathbf{x})^{-1}}{\lambda(\mathbf{x})^{-1} + \frac{P(H_1)}{P(H_0)}} \tag{3.4b}$$

Hence, a decision on a hypothesis can be made in either of the domains with equivalent results: based on a threshold of the likelihood ratio or the corresponding threshold of the Bayesian probability.

The sequential likelihood ratio test (SLRT) reformulates the LRT as a sequential analysis problem with a dynamic data collection process [96]. Given a sequence of observations, $\mathbf{x}_t = [x_1, x_2, \dots, x_t]$, the goal of the SLRT is to decide which hypothesis is correct within a desirable level of error in as few observations as possible. The cumulative likelihood ratio, $\Lambda(\mathbf{x}_t)$, is evaluated based on the ratio of joint likelihood of the observations, $p(x_1, x_2, \dots, x_t|H_i)$:

$$\Lambda(\mathbf{x}_t) = \frac{p(x_1, x_2, \dots, x_t|H_1)}{p(x_1, x_2, \dots, x_t|H_0)} \tag{3.5}$$

For simplicity, $\Lambda(\mathbf{x}_t)$ and $\lambda(\mathbf{x}_t)$ will be denoted as Λ_t and λ_t , respectively. Decision regions for Λ_t are specified to either continue data collection or stop data collection and select one of the hypotheses based on specified thresholds:

$$\Lambda_t \begin{cases} \leq \mathcal{A} \Rightarrow \text{stop, choose } H_0 \\ \geq \mathcal{B} \Rightarrow \text{stop, choose } H_1 \\ \in (\mathcal{A}, \mathcal{B}) \Rightarrow \text{take another sample} \end{cases} \tag{3.6}$$

where \mathcal{A} and \mathcal{B} are the decision thresholds. The threshold bounds are chosen based on

specified probabilities of detection, $P_D = P(\text{deciding } "H_1" | H_1 \text{ is the true hypothesis})$ and false alarm, $P_{FA} = P("H_1" | H_0)$:

$$\mathcal{A} = \frac{1 - P_D}{1 - P_{FA}} \quad (3.7a)$$

$$\mathcal{B} = P_D / P_{FA} \quad (3.7b)$$

Equivalently, \mathcal{A} and \mathcal{B} can be specified in terms of class conditional error levels, α and β , where $\alpha = P_{FA}$ and $\beta = 1 - P_D$. Likewise, Bayes' rule can be implemented sequentially, with data collection stopped when a hypothesis' posterior probability attains a preset threshold value prior to decision making, as is the case with the Bayesian DS algorithm.

The performance of the SLRT is characterised by the operating characteristic (OC) and the average sample number (ASN) functions. The OC function, $\mathcal{L}(\theta_i)$, is the probability of terminating the test with an H_0 decision, given that H_i is the true hypothesis. The bounds of $\mathcal{L}(\theta_i)$ are determined by $1 - \alpha$ and β , where H_0 and H_1 , respectively, are the true hypotheses. Also, the average probability of error, p_e , is based on the class conditional error rates and the hypothesis priors:

$$p_e = \alpha P(H_0) + \beta P(H_1) \quad (3.8)$$

The ASN function is the average number of observations prior to termination of the test. For the SLRT, if the observations are assumed to be independent and identically distributed (i.i.d.), the cumulative likelihood ratio (3.5) can be expressed as the product of the individual likelihood ratios of the observations, or as a logarithmic sum, for convenience:

$$\Lambda_t = \prod_{\tau=1}^t \frac{p(x_\tau|H_1)}{p(x_\tau|H_0)}$$

$$\ln \Lambda_t = \sum_{\tau=1}^t \ln \lambda_\tau \quad (3.9)$$

The ASN for the SLRT in an i.i.d. case can be approximated by exploiting Wald's equation [97] (Appendix A). Since $\ln \Lambda_t$ can be expressed as a random sum of $\ln \lambda_\tau$'s, the ASN under a given hypothesis can be estimated:

$$\mathbb{E}[T|H_i] = \frac{\mathbb{E}[\ln \Lambda_t|H_i]}{\mathbb{E}[\ln \lambda_t|H_i]} \quad (3.10a)$$

$$\mathbb{E}[T|H_1] = \frac{P_D \ln \mathcal{B} + (1 - P_D) \ln \mathcal{A}}{\mathbb{E}[\ln \lambda_t|H_1]} \quad (3.10b)$$

$$\mathbb{E}[T|H_0] = \frac{(1 - P_{FA}) \ln \mathcal{B} + P_{FA} \ln \mathcal{A}}{\mathbb{E}[\ln \lambda_t|H_0]} \quad (3.10c)$$

where $\mathbb{E}[T|H_i]$ is the ASN; $\mathbb{E}[\ln \Lambda_t|H_i]$ is the expected cumulative log-likelihood ratio (CLLR) when data collection is stopped; and $\mathbb{E}[\ln \lambda_t|H_i]$ is the expected log-likelihood (LLR), given H_i is the true hypothesis.

In real world testing, it should be noted that α and β will represent the upper bounds for the error rates of the OC function. Also, (3.10) represents the asymptotic lower bound of the ASN. These performance estimates omit the tendency of $\ln \Lambda_t$ to overshoot or undershoot the threshold bounds. As a result, the actual error levels will be smaller than α and β , and the ASN will be higher than that specified by Wald's equation. For the SLRT with i.i.d. observations, better approximations of the performance functions can be obtained by solving a series of inductive integral equations governing the OC and the ASN functions [98].

In the Bayesian DS algorithm as implemented in [25], the classifier score observations are considered to be independent. The equivalent likelihood formulation of

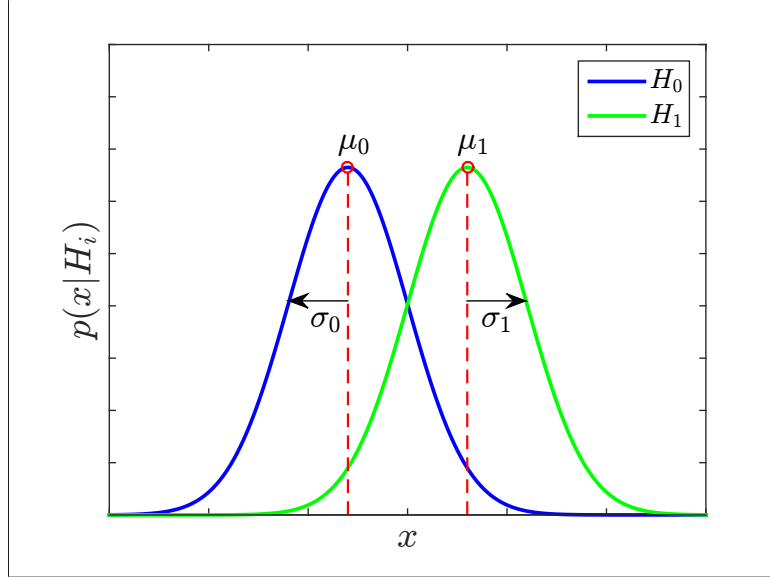


FIGURE 3.1: Illustrative example of the class conditional probability distribution parameters, (μ_0, σ_0^2) and (μ_1, σ_1^2) , used to determine the detectability index of two normal distributions.

the Bayesian probabilities for each character can be expressed as a sum to derive the performance functions of the Bayesian DS algorithm. A corresponding likelihood ratio formulation for the Bayesian DS algorithm is also more convenient as it avoids having to analyse the more complex denominator term in the Bayesian update equation (2.9).

The performance of a hypothesis test depends on how discriminable the probability models under each hypothesis are from each other. The detectability index, d , is a measure that quantifies the discriminability between two normal probability density functions by their separation and spread [55]. For two normally distributed random variables, the detectability index is computed from the class conditional means and standard deviations, (μ_0, σ_0^2) and (μ_1, σ_1^2) , as shown in Figure 3.1:

$$d = \frac{|\mu_1 - \mu_0|}{\sqrt{0.5(\sigma_1^2 + \sigma_0^2)}} \quad (3.11)$$

The detectability index is used to parameterise the probability distribution of $\ln \lambda$ under each hypothesis [44]:

$$\ln \lambda \sim \begin{cases} \mathcal{N}\left(\frac{-d^2}{2}, d^2\right) | H_0 \\ \mathcal{N}\left(\frac{d^2}{2}, d^2\right) | H_1 \end{cases} \quad (3.12)$$

Hence, within the P300 speller context, operating in the likelihood ratio domain provides a convenient way to quantify a user’s performance level via the P300 classifier discriminability. For example, given a user’s training data, the respective means and variances of the classifier likelihoods, $p(y|H_0)$ and $p(y|H_1)$, can be used to determine on average how a user will perform with the system. To account for changes in performance level from that achieved during calibration, it is also easy to visualise and study performance in a 1-D space through different values of d , as opposed to a 4-D space, $\{\mu_0, \sigma_0^2, \mu_1, \sigma_1^2\}$, as would be required using the method described in [53].

3.1.3 *Deriving Performance Estimates for the Bayesian Dynamic Stopping Algorithm*

Performance measures have been studied for the binary SLRT [96], as well as M -ary hypothesis tests [99, 100, 101, 102]. However, additional considerations are required when estimating performance of the Bayesian DS algorithm within the P300 speller context. Most sequential tests assume the availability of an infinite amount of time to make decisions, which is not practical for real world applications. Also, most sequential tests assume that each hypothesis represents a distinct probability model and the observations are i.i.d.

For this application, the classifier scores in the Bayesian DS algorithm are assumed to be independent but not identically distributed as they are generated as a result of alternative random presentation of non-target and target stimulus events.

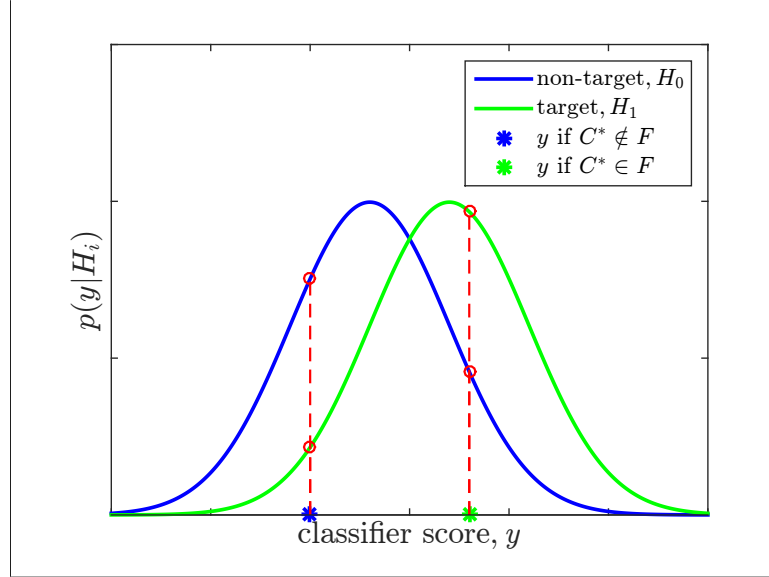


FIGURE 3.2: Role of probability density functions (pdfs) during the probability update process in the Bayesian dynamic stopping algorithm. The pdfs for non-target and target classifier scores represent the H_0 and H_1 hypotheses, respectively. A flashed subset, \mathcal{F} , that contains the target character, C^* , should elicit a P300, resulting in a higher classifier score such that $p(y|H_1) > p(y|H_0)$. A flashed subset that does not contain the target character should not elicit a P300, resulting in a lower score such that $p(y|H_1) < p(y|H_0)$.

If the stimulus flashes are a subset of characters, each observation is technically generated from a subset of the hypotheses. Even with single character presentation, the probability model of a given character depends on whether it is or is not the target character. Nonetheless, with these considerations, the P300 speller model with the Bayesian DS algorithm can be simplified from an M -ary to a binary hypothesis problem, and the likelihood ratio test exploited to derive the performance measures for the DS algorithm when considered in terms of the discriminability of the class conditional pdfs.

The importance of the discriminability of the class conditional pdfs in the Bayesian update process can be visualised by considering two cases of flashed character subsets that include or do not include the target character. Figure 3.2 illustrates two scenarios of classifier scores produced in the presence/absence of the target character in a

flashed subset. If the target character is in the flashed subset, $C^* \in \mathcal{F}_t$, a P300 signal will likely be elicited, resulting in a high classifier score, hence $p(y_t|H_1) > p(y_t|H_0)$ or $\lambda > 1$. Conversely, if $C^* \notin \mathcal{F}_t$, it is likely that $p(y_t|H_1) < p(y_t|H_0)$ or $\lambda < 1$. In both cases, the target probability is updated with a high likelihood value, i.e. $p(y_t|C_m = C^*, \mathcal{F}_t)$ according to (2.10), hence the target character probability should increase progressively towards 1.

Consequently, the P300 speller system can also be viewed as evaluating M different binary hypothesis tests. Each time a character is flashed, the BCI evaluates its CLLR to determine if it is or is not the target character (3.9):

$$L_m(T) = \sum_{t=1}^T l_m(t) \mathbb{1}_{\{C_m \in \mathcal{F}_t\}} \quad (3.13)$$

$L_m(T)$ is the CLLR for character C_m ; $l_m(t) = \ln \left(\frac{p(y_t|H_1)}{p(y_t|H_0)} \right)$ is the observation LLR for character C_m at the t^{th} stimulus event based on the classifier score, y_t ; and $\mathbb{1}_{\{C_m \in \mathcal{F}_t\}}$ is an indicator function, which equals 1 if $C_m \in \mathcal{F}_t$, and is 0 otherwise. With data collection, $L_m(T)$ for the target character should increase progressively towards the upper threshold of the LRT, while those of the non-target characters should decrease.

The random variable, $L_m(T)$, can now be analysed to estimate system performance. Since $L_m(T)$ is a sum of random variables, its probability distribution can be determined by convolution, with the probability distribution of $l_m(t)$ defined according to the presence/absence of a target character in a flash group. If the classifier likelihoods are normally distributed, then $l_m(t)$ and $L_m(T)$ are also normally distributed and parameterised by d (3.12).

It should again be noted that sequential tests assume the availability of an infinite amount of time to make a decision. However, for real world applications, a data truncation limit, t_{max} , is usually imposed for practical reasons. A sequential test with

a data collection limit will be referred to as *truncated*, and that with no data collection limit as *untruncated*. The class conditional probability of errors, α and β define the maximum error levels of the untruncated SLRT. The stopping threshold probability, P_{th} , defines the minimum accuracy level that can be achieved with an untruncated Bayesian DS algorithm. Data truncation will affect both a test's error level and the ASN. For example, with the SLRT, $\ln \Lambda_t$ might not necessarily converge to either of the threshold values by t_{\max} . As a result, a truncated sequential test may achieve a higher probability of error than the maximum level guaranteed by an untruncated test [103]. Appendix D details the derivation of the performance measures in the case of a truncated SLRT. A similar procedure can be used to determine the equivalent performance measures based on the CLLR of each character. The imposition of a data truncation limit is thus an important consideration when deriving performance estimates for the Bayesian DS algorithm, as implemented within the P300 speller context.

Accuracy

To determine the projected accuracy for the Bayesian DS algorithm, the probability that the BCI correctly selects the target character, $P(\hat{C} = C_i | C_i = C^*)$, at the time that data collection is stopped has to be evaluated. This probability is averaged across all the characters in the grid:

$$\begin{aligned}
 A &= \sum_{i=1}^M P(\hat{C} = C_i | C_i = C^*) P(C_i) \\
 &= \sum_{i=1}^M P\left(\max_{j \neq i} L_j(t_s) < L_i(t_s) | C_i = C^*\right) P(C_i) \tag{3.14}
 \end{aligned}$$

where $P\left(\max_{j \neq i} L_j(t_s) < L_i(t_s) \mid C_i = C^*\right)$ is the probability that the CLLR of character, C_i , exceeds the CLLRs of the rest of the characters, given that $C_i = C^*$; and $P(C_i)$ is the probability that character C_i can be the target character.

From (3.14), let $Y_i = L_i(t_s)$ and $X_i = \max_{j \neq i} L_j(t_s)$, given $C_i = C^*$. The probability, $P(X_i < Y_i)$ can be determined based on random inequalities (Appendix B):

$$P(X_i < Y_i) = \int_{-\infty}^{\infty} F_{X_i|Y_i=z}(z) f_{Y_i}(z) dz \quad (3.15)$$

where f_{Y_i} is the pdf of Y_i , and $F_{X_i|Y_i}$ is the conditional cumulative distribution function (cdf) of X_i given Y_i . Determining the probability distribution of the maximum of N random variables, i.e. X_i , is a problem considered in order statistics (Appendix C).

For a dynamic stopping algorithm, t_s is a random variable, with its maximum value capped by the data collection limit, i.e. $1 \leq t_s \leq t_{\max}$. Deriving the accuracy for the truncated Bayesian DS algorithm from an equivalent truncated test in the likelihood ration domain entails analysing the CLLRs assuming that data collection stops at any value in the range, $1, 2, \dots, t_{\max}$, with an associated probability for each stopping time. This requires numerical analysis, which can be computationally expensive if t_{\max} is large.

For simplicity, the accuracy will instead be derived for a Bayesian algorithm with fixed sample size (FSS), where the stopping time is fixed, i.e. $t_s = t_{\max}$. With a dynamic stopping algorithm, i.e. when a threshold probability is imposed, not all of the flash groups defined within the sequence limit, t_{\max} , may necessarily be illuminated prior to stopping data collection. Nonetheless, it is hypothesised that estimating the accuracy for a Bayesian FSS test with data collection limit, t_{\max} , can provide a good estimate for the underlying system accuracy of the Bayesian DS

algorithm with the same data collection limit.

For a given stimulus paradigm, if the base pattern is replicated based on the number of sequences, s , the average CLLR across sequences can be considered, $\bar{L}_m(T)$, or equivalently the sequence average of the observation LLR, $\bar{l}_m(t)$:

$$\begin{aligned}
 \bar{L}_m(t_b) &= \frac{1}{s} \sum_{n=1}^s L_{m,n}(t_b) \\
 &= \sum_{t=1}^{t_b} \frac{1}{s} \sum_{n=1}^s l_{m,n}(t) \mathbb{1}_{\{C_m \in \mathcal{F}_t\}} \\
 &= \sum_{t=1}^{t_b} \bar{l}_m(t) \mathbb{1}_{\{C_m \in \mathcal{F}_t\}}
 \end{aligned} \tag{3.16}$$

where t_b is the number of flashes within a sequence of a stimulus paradigm with a base pattern; $\bar{L}_m(t_b)$ is the sequence average CLLR for character, C_m ; and $\bar{l}_m(t)$ is the sequence average of the observation LLR at time index, t . For example, for the RCP of size $R \times C$, $t_b = R + C$, and with s sequences, $t_{\max} = s \times (R + C)$

Expected Stopping Time

The expected stopping time, EST, is the average number of stimulus event presentations required for any character to reach the probability threshold value prior to character selection. The EST for the Bayesian DS algorithm can be derived from the equivalent LLR formulation. Since the BCI eventually selects one of the characters as the user's intended target character, it is only necessary to analyse the rate of convergence of the CLLRs, $L_m(t)$, to the stopping threshold given that the target character is one of the M characters, i.e. under the condition that the H_1 hypothesis is always true. The first step requires determining the average number of times the selected flash group is sampled prior to reaching the stopping threshold. Then, taking into account the average character-to-character interval, the effective number

of stimulus event presentations prior to decision making can be determined.

However, just like with accuracy, there are several considerations involved in determining the EST of a truncated test. The data collection limit determines the maximum number of stimulus events the BCI system is allowed to present prior to decision making. Due to this constraint, the probability distribution of the BCI system stopping data collection at the various stopping times, from $1, 2, \dots, t_{max}$, has to be determined to derive the EST (Appendix D). Alternatively, the imposition of a data collection limit can be relaxed to instead derive the asymptotic lower bound of the EST for an untruncated DS algorithm. An additional issue is that the observation LLRs for each character, $l_m(t)$, are only i.i.d. for the target character, and non-target characters that are not in the same flash group as the target character. Nonetheless, under certain stimulus paradigm conditions, the observations are assumed to be i.i.d. for both the target and all of the non-target characters by conditioning on a specific target character, e.g. the RCP. In such a scenario, the EST can be determined by exploiting Wald's equation, in a manner similar to that presented in (3.10b).

3.2 Methods

There are two methods of obtaining the BCI performance measures: analytically by analysing the cumulative response function, $\Theta_m(t)$, or via Monte Carlo simulations. In this section, both methods will be described for the Bayesian algorithm based on the equivalent CLLR formulation, $L_m(t)$.

3.2.1 Analytical Estimation

(i) Row-column paradigm

For the row-column paradigm, the BCI correctly selects the target character if the following event occurs: $\{\text{mean CLLR of the target row} > R - 1 \text{ mean CLLRs of the non-target rows}\} \cap \{\text{mean CLLR of the target column} > C - 1 \text{ mean CLLRs of the}$

non-target columns}. Due to this property, the performance for selecting a target row can be analysed separately from that of selecting the target column, and combined accordingly to derive the overall performance measure. Due to symmetry of the RCP, it is only necessary to determine a performance measure based on considering one character as the target character, instead of averaging over all characters, as in (3.14). Also, since the row/column flash groups are pairwise disjoint, the cumulative classifier scores associated with each group during row/column selection are independent of each other. This leads to tractable solutions for the performance estimates of the RCP.

Accuracy

Determining the system accuracy for the RCP requires evaluating the probability that the BCI correctly selects both the target row and target column. From (3.14-3.15), the projected accuracy is determined according to:

$$\begin{aligned}
 A_{pr} &= p_r \times p_c \\
 &= \int_{-\infty}^{\infty} (F_{\bar{U}}(z))^{R-1} f_{\bar{T}}(z) dz \times \int_{-\infty}^{\infty} (F_{\bar{U}}(z))^{C-1} f_{\bar{T}}(z) dz \quad (3.17)
 \end{aligned}$$

where p_r denotes the probability of correctly selecting the target row; p_c is the probability of correctly selecting the target column; \bar{T} denotes the average CLLR for the target row/column; \bar{U} denotes the average CLLR for the non-target rows/columns; R and C denote the number of rows and columns, respectively, in the grid; and $F_X(\cdot)$ and $f_X(\cdot)$ denote the cumulative density function and probability density function, respectively, for the random variable X .

For the RCP, the projected accuracy can be determined given the grid size, the pdfs of the non-target and target classifier scores, and the maximum number of sequences. The pdfs of $\ln \lambda$ can be derived from the pdfs of the non-target and the target classifier scores, $p(y|H_0)$ and $p(y|H_1)$, respectively. The pdfs of \bar{U} and \bar{T} can

Algorithm 1: Projected Accuracy for the Row-Column Paradigm

Input: d = detectability index; R = number of rows; C = number of columns; s = maximum number of sequences
Output: A_{pr} = Projected accuracy
1 Function rcpAccuracy(d, R, C, s)
2 $\mu_0 = -d^2/2$
3 $\mu_1 = d^2/2$
4 $\sigma = d/\sqrt{s}$
5 $\triangleright \Phi(\cdot)$ and $\phi(\cdot)$ are cdf and pdf, respectively, of the standard normal distribution
6 foreach $j = \{r, c\}$ **do**
7 $p_j = \int_{-\infty}^{\infty} \left\{ \Phi\left(\frac{t-\mu_0}{\sigma}\right) \right\}^{J-1} \frac{1}{\sigma} \phi\left(\frac{t-\mu_1}{\sigma}\right) dt$ $\triangleright J = \{R, C\}$
8 end
9 return $A_{pr} = p_r \times p_c$

be derived via numerical convolution of the pdfs of $\ln \lambda$, under the non-target and target hypothesis cases, respectively. If the classifier scores are normally distributed, the pdf of $\ln \lambda$ specified by the detectability index (3.12) can be used to derive the pdfs of \bar{T} and \bar{U} . Under a normality assumption, Algorithm 1 outlines pseudo-code to obtain the projected accuracy with the Bayesian FSS algorithm, which can be used to approximate the accuracy with a truncated Bayesian DS algorithm given the same data collection limit.

Expected Stopping Time

Similarly, for the EST, the average number of stimulus event presentations prior to making a row selection decision, is evaluated separately from the average number of flashes prior to a column selection decision. The asymptotic lower bound for the EST for the untruncated DS algorithm can easily be derived from the equivalent CLLR formulation of the Bayesian DS algorithm by evaluating M binary SLRTs (3.9) until a row or column selection decision is made.

First, the average number of times a row or column is flashed has to be determined, given that the target row or column is always among the flash groups. The stopping thresholds for the CLLR, $L_m(t)$, of the row and column selection process can be derived from the Bayesian probability threshold, P_{th} according to (3.4a)-(3.4b)

[102]. The overall sequential test only terminates when the CLLR of a row/column flash groups attains the upper stopping threshold, and that flash group is declared as the target row/column. Hence it is only required to evaluate the ASN for the corresponding SLRT given the H_1 hypothesis is true.

Taking into account the average character-to-character interval within a row or column sequence, which is equivalent to the number of rows or columns in the case of the RCP, the asymptotic lower bound of the expected stopping time, EST_{as} , can be determined:

$$EST_{as} = R\mathbb{E}[T_r|H_1] + C\mathbb{E}[T_c|H_1] \quad (3.18)$$

where R/C is the number of rows/columns in the grid; $\mathbb{E}[T_r|H_1]$ or $\mathbb{E}[T_c|H_1]$ is the ASN for a row/column SLRT, given that a target row/column is always present.

For the RCP, given the grid size, pdfs of the non-target and target classifier scores and probability threshold, EST_{as} can be determined accordingly, from the pdfs of $\ln \lambda$. If the classifier scores are normally distributed, the detectability index can be used to quantify EST_{as} via $\mathbb{E}[\ln \lambda|H_1]$ (3.12). Under a normality assumption, Algorithm 2 outlines pseudo-code to determine the asymptotic lower bound of the EST with the untruncated Bayesian DS algorithm.

Similar Block-randomised Paradigms

The properties of the RCP allow for the derivation of performance measures analytically with tractable solutions. These analytical methods can also be applicable to other similar block-randomised stimulus paradigms that satisfy the general properties of the RCP:

1. The stimulus paradigm renders P300 speller selection in two stages: selection of a flash group from a set of pairwise disjoint flash groups, $\mathcal{F}^1 = \{\mathcal{F}_i^1\}_{i=1}^R$, followed by selection from another set of pairwise disjoint flash groups, $\mathcal{F}^2 =$

Algorithm 2: Asymptotic Lower Bound of the Expected Stopping Time for the Row-Column Paradigm

Input: d = detectability index; R = number of rows; C = number of columns; P_{th} = stopping probability threshold
Output: EST_{as} = Asymptotic lower bound for the expected stopping time

```

1 Function rcpESTAsymptote( $d, R, C, P_{th}$ )
2  $A_0 = A_1 = (1 - P_{th})/P_{th}$ 
3  $\mathbb{E}[\ln \lambda | H_1] = d^2/2$ 
4 foreach  $j = \{r, c\}$  do
5      $\pi_1(j) = 1/J$ 
6      $\pi_0(j) = 1 - \pi_1(j)$ 
7      $A_j^* = \pi_0(j)A_0/\pi_1(j)$ 
8      $B_j^* = \pi_0(j)/(A_1\pi_1(j))$ 
9      $\mathbb{E}[T_j | H_1] = \frac{P_{th} \ln B_j^* + (1 - P_{th}) \ln A_j^*}{\mathbb{E}[\ln \lambda | H_1]}$ 
10 end
11 return  $EST_{as} = R\mathbb{E}[T_r | H_1] + C\mathbb{E}[T_c | H_1]$ 

```

$\{\mathcal{F}_j^2\}_{j=1}^C$. For two flash groups from each set, \mathcal{F}_i^1 and \mathcal{F}_j^2 , if $\mathcal{F}_i^1 \cap \mathcal{F}_j^2 \neq \emptyset$, then $|\mathcal{F}_i^1 \cap \mathcal{F}_j^2| = 1$, $\mathcal{F}_i^1 \cap \mathcal{F}_n^2 = \emptyset$, $\forall n \neq j$, and $|\mathcal{F}_m^1 \cap \mathcal{F}_j^2| = \emptyset$, $\forall m \neq i$. For the RCP, each row shares one common character with only one of the column flash groups.

2. All of the flash groups in each selection stage are equally sized, i.e. $|\mathcal{F}_i^1| = K$ and $|\mathcal{F}_j^2| = L$. For the RCP, there are same number of characters in each row or column group.
3. Within each sequence, the flash groups are sampled without replacement. There is no randomisation of character-to-flash group assignment after each sequence. For the RCP, the characters in each row or column are fixed, with the order of presentation randomised without replacement after each sequence.

It should also be noted that a P300 speller with single character presentation lends itself to deriving the performance measures analogous to selecting from only one flash group sub-set, i.e. in Algorithms 1-2, $C = 0$.

(ii) *Generic Stimulus Paradigms*

For most stimulus paradigms, a base stimulus pattern is usually defined, replicated based on the number of sequences, and the character-to-flash group assignment is randomised within each sequence according to the paradigm randomisation rules. Some stimulus paradigms used in the BCI literature do not satisfy the general characteristics of the RCP e.g. [50, 71, 104]. Therefore the proposed methods outlined in Algorithms 1-2 for estimating performance cannot be used.

Evaluating the performance for these stimulus paradigms may still be achieved by analysing the corresponding CLLRs, $L_m(t)$, if the flash pattern of the stimulus paradigm is known. The projected accuracy for stimulus paradigm can be derived from the general expression in (3.14). For those algorithms that incorporate character-to-flash group randomisation within each sequence, the uncertainty due to the flash group assignments has to be accounted for. For example, it is no longer possible to derive an estimate for accuracy using the sequence average of the LLR, $\bar{L}_m(T)$, as in (3.16).

Nonetheless, if it is assumed that the characters within flash groups are fixed, i.e. there is no character-to-flash group randomisation, it is possible to obtain an estimate for the accuracy of a given paradigm using $\bar{L}_m(T)$ (3.16). This estimate will also specify a lower bound for the projected accuracy for a given stimulus paradigm; this is because character-to-codeword randomisation can minimise the correlation between the character CLLRs as more data is collected.

For a generic stimulus paradigm, the projected accuracy can be approximated given the sequence flash pattern, the pdfs of the non-target and target classifier scores and the maximum number of sequences. If the classifier scores are normally distributed, the pdf of $\ln \lambda$ can be used to derive the pdfs for $\bar{L}_m(T)$ according to (3.12). Under a normality assumption, Algorithm 3 outlines pseudo-code to deter-

Algorithm 3: Projected Accuracy for a Generic Stimulus Paradigm

Input: d = detectability index; \mathcal{F} = sequence flash pattern matrix; s = maximum number of sequences;
 $\{P(C_m)\}_{m=1}^M$ = character probability distribution, $\{C_m\}_{m=1}^M$

Output: A_{pr}^* = Generalised projected accuracy

- 1 **Function** genParadigmAccuracy($d, \mathcal{F}, s, \{P(C_m)\}_{m=1}^M$)
- 2 $[M, T] = \text{size}(\mathcal{F})$ ▷ M = number of grid characters, T = number of flashes/sequence
- 3 **for** $i = 1 : M$ **do**
- 4 ▷ C_i is the target character
- 5 $\bar{l}_i(t) \sim \mathcal{N}(d^2/2, d/\sqrt{s})$
- 6 $\bar{L}_i(T) = \sum_{t=1}^T \bar{l}_i(t) \mathbb{1}_{\{C_i \in \mathcal{F}_t\}}$
- 7 ▷ Let $Y = \bar{L}_i(T)$
- 8 **for** $\forall j \neq i$ **do**
- 9 $\bar{l}_j(t) \sim \begin{cases} \mathcal{N}(d^2/2, d/\sqrt{s}), C_i \notin \mathcal{F}_t \\ \mathcal{N}(-d^2/2, d/\sqrt{s}), C_i \in \mathcal{F}_t \end{cases}$
- 10 $\bar{L}_j(T) = \sum_{t=1}^T \bar{l}_j(t) \mathbb{1}_{\{C_j \in \mathcal{F}_t\}}$
- 11 **end**
- 12 - Evaluate covariance matrix, $[\bar{L}_1(T), \dots, \bar{L}_M(T)]$
- 13 - Evaluate probability distribution of $X = \max_{\forall j \neq i} \bar{L}_j(T)$ ▷ (Appendix C)
- 14 - Determine correlation $\rho_{X,Y}$ ▷ (Appendix C)
- 15 $P(\hat{C} = C_i | C_i = C^*) = \int_{-\infty}^{\infty} F_{X|Y=z}(z) f_Y(z) dz$ ▷ (Appendix B)
- 16 **end**
- 17 **return** $A_{pr}^* = \sum_{m=1}^M P(\hat{C} = C_i | C_i = C^*) P(C_i)$

mine the projected accuracy with the Bayesian FSS algorithm, as well as approximate that with a truncated Bayesian DS algorithm given the same data collection limit.

3.2.2 Monte Carlo Estimation

Alternatively, performance estimates for any stimulus paradigm can be obtained from Monte Carlo simulations [45]. This is particularly useful in circumstances where a performance measure cannot easily be obtained by analytical means or from numerical evaluation of a closed-form expression.

In general, to simulate P300 speller runs, the following elements are required:

- The stimulus paradigm flash pattern, \mathcal{F} , which determines the characters in each flash group and the order of presentation of the flash groups.
- The class conditional classifier likelihood functions, $p(y|H_0)$ and $p(y|H_1)$, which determine the random sampling of the classifier score in the case of a non-target

response or a target EEG response, respectively.

- The character cumulative classifier response function, $\Theta_m(t)$, which quantifies the likelihood that each character is the target character, given all of the user's EEG responses.
- A stopping rule, static or dynamic, to terminate data collection and a decision rule based on $\Theta_m(t)$ to decide on a character selection, \hat{C} .

Algorithm 4 outlines pseudo-code for simulating P300 speller character selection for a generic target character estimation algorithm with a given stimulus paradigm, in order to estimate accuracy and the expected stopping time. Since performance profile curves are obtained via maximum likelihood estimation, a high number of iterations is desirable to obtain a low variance on the estimates [43, 45]. For this study, P300 speller simulations were performed with the Bayesian probability update process as the cumulative classifier response function. Also, the classifier likelihood functions, $p(y|H_0)$ and $p(y|H_1)$, were normally distributed with parameters (μ_0, σ_0^2) , (μ_1, σ_1^2) , specified accordingly based on a given detectability index.

3.3 Results

3.3.1 Verification of the Theoretical Model

Monte Carlo simulations of P300 spelling runs (Algorithm 4) were performed to verify the performance measures obtained analytically from the equivalent likelihood ratio formulation of the Bayesian DS algorithm. A performance measure is described, Γ_j^i , where $\Gamma = \{A, EST\}$ is the type of performance measure, $i = \{a, s\}$ denotes whether the measure was determined analytically (a) or via simulations (s); and $j = \{fss, tds, uds, as\}$, denotes the type of Bayesian algorithm, fixed sample size (fss), truncated DS (tds), untruncated DS (uds), and an asymptotic lower bound (as) in the case of untruncated DS.

Algorithm 4: Simulation of P300 Speller Character Selection to Estimate Performance Measures

Input: N = Number of iterations; $\{P(C_m)\}_{m=1}^M$ = character probability distribution; $\{p(y|H_0), p(y|H_1)\}$ = class conditional classifier score likelihoods; \mathcal{F} = sequence flash pattern; s = maximum number of sequences; (if applicable) Θ_{th} = stopping threshold

Output: \hat{A} = Accuracy, \hat{EST} = Expected stopping time

- 1 **Function** simulateP300Speller($N, \{P(C_m)\}_{m=1}^M, \{p(y|H_0), p(y|H_1)\}, \mathcal{F}, s, \Theta_{th}$)
- 2 $[M, T] = \text{size}(\mathcal{F})$ \triangleright M = number of grid characters, T = number of flashes/sequence
- 3 $t_{\max} = s \times T$
- 4 **for** $i = 1 : N$ **do**
- 5 - Randomly draw from $\{P(C_m)\}_{m=1}^M$ for a target character, C_i^*
- 6 - Generate new flash pattern matrix, \mathbb{F} , of size $[M, t_{\max}]$, by replicating and randomising \mathcal{F} according to paradigm randomisation rules
- 7 - Initialise character cumulative classifier response function, $\{\Theta_m(0)\}_{m=1}^M$
- 8 - $t \leftarrow 0$
- 9 **while** $\max \Theta_m(t) < \Theta_{th}$ and/or $t < t_{\max}$ **do**
- 10 - $t \leftarrow t + 1$
- 11 - Sample flash group, \mathbb{F}_t
- 12 - Randomly draw classifier score, y_t , according to $\begin{cases} p(y_t|H_0), C_i^* \notin \mathbb{F}_t \\ p(y_t|H_1), C_i^* \in \mathbb{F}_t \end{cases}$
- 13 - Update $\{\Theta_m(t)\}_{m=1}^M$, according to algorithm update rules
- 14 **end**
- 15 $\hat{C}_i = \underset{m}{\operatorname{argmax}} \Theta_m(t)$
- 16 $nFlash_i = t$
- 17 **end**
- 18 **return** $\hat{A} = \frac{1}{N} \sum_{i=1}^N eq(\hat{C}_i, C_i^*), \hat{EST} = \frac{1}{N} \sum_{i=1}^N nFlash_i$

The performance measures obtained analytically, A_{fss}^a and EST_{as}^a , were determined from Algorithm 1 or 3, and 2, respectively. Using normal distributions, P300 spelling runs were simulated according to a given stimulus paradigm over a range of detectability indices. In general, three types of simulations were performed:

- Simulation I: The Bayesian algorithm with a fixed sample size (FSS), with a decision rule, $\hat{C} = \underset{m}{\operatorname{argmax}} P_m(t_{\max})$ to determine accuracy, A_{fss}^s .
- Simulation II: The truncated Bayesian DS algorithm with a decision rule, $\hat{C} = \underset{m}{\operatorname{argmax}} P_m(t_s) \geq P_{th}$, with $1 \leq t_s \leq t_{\max}$, to determine accuracy, A_{tds}^s and expected stopping time, EST_{tds}^s .
- Simulation III: The untruncated Bayesian DS algorithm with a decision rule,

$$\hat{C} = \underset{m}{\operatorname{argmax}} P_m(t_s) \geq P_{th}, \text{ with } 1 \leq t_s \leq \infty, \text{ to determine } A_{uds}^s \text{ and } EST_{uds}^s.$$

Figures 3.3 and 3.4 show the results for the accuracy and the expected stopping time, respectively, for the RCP with a 9×8 grid. Figure 3.3(a) compares the projected accuracy obtained analytically, A_{fss}^a (Algorithm 1), and the accuracy obtained by simulations, A_{fss}^s , with the Bayesian FSS algorithm, demonstrating a near-identical match (mean square error, $MSE = 2.1 \times 10^{-5}$). This confirms that the P300 speller with the Bayesian dynamic stopping algorithm can be modeled with an equivalent likelihood ratio formulation, and in the likelihood domain, performance can conveniently be parameterised by the detectability index.

Figure 3.3(b) compares the effect on accuracy of imposing a probability threshold on the Bayesian algorithm to achieve dynamic stopping. In the Bayesian DS algorithm, a threshold, $P_{th} = 0.9$, was imposed. The accuracy of the truncated DS algorithm, A_{tds}^s is still very similar to the analytical estimate of the FSS algorithm, A_{fss}^a ($MSE = 1.59 \times 10^{-2}$). In general, accuracy tends to improve with increased data collection. Given the same data collection limit, there is a potential loss in accuracy that can result with a dynamic stopping algorithm due to the varied stopping time, $1 \leq t_s \leq t_{\max}$, as opposed to using all of the data allowed by the data collection limit, $t_s = t_{\max}$, as is the case with a FSS algorithm. At higher detectability indices, it can be observed that there are slight deviations of A_{tds}^s from A_{fss}^a , since it is more likely that the expected stopping time for the truncated DS algorithm will be less than t_{\max} .

The accuracy of the truncated Bayesian DS algorithm at a given detectability index is upper bounded by the accuracy of the untruncated algorithm. The untruncated DS algorithm will achieve accuracy levels equal to or greater than P_{th} . From Figure 3.3(b), it can be observed that $A_{uds}^s \geq 0.9$ and A_{uds}^s defines the maximum value attainable by A_{tds}^s . Hence, $A_{fss}^a = A_{tds}^s$ if $A_{fss}^a \lesssim P_{th}$, and $A_{fss}^a > A_{tds}^s$ if $A_{fss}^a > P_{th}$.

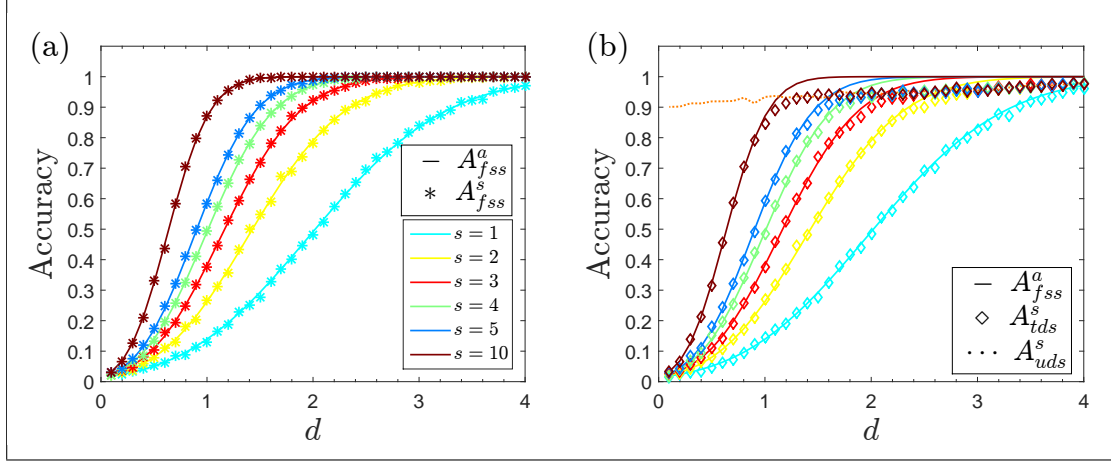


FIGURE 3.3: Accuracy, A , vs. detectability index, d , with the Bayesian algorithm for a row-column paradigm with a 9×8 grid. P300 spelling runs with the Bayesian algorithm were simulated using normal distributions (for target and non-target likelihoods) and a sequence limit, s . The maximum number of flashes, $t_{\max} = s \times (9 + 8)$ flashes. (a) Comparison of the accuracy obtained analytically, A_{fss}^a (Algorithm 1) to that determined via simulations, A_{fss}^s , for the Bayesian stopping algorithm with fixed sample size (FSS). (b) Comparison of A_{fss}^a , to that determined via simulations for the truncated, A_{tds}^s and untruncated, A_{uds}^s , Bayesian dynamic stopping algorithm with probability threshold, $P_{th} = 0.9$.

The range of detectability indices over which the difference, $|A_{fss}^a - A_{tds}^s| \leq 1 - P_{th}$, is appreciably minimised if P_{th} is closer to 1. Hence, the projected accuracy for a Bayesian FSS algorithm can be used to approximate that of the truncated Bayesian DS algorithm given the same data collection limit, with the accuracy of the truncated algorithm upper-bounded at around P_{th} .

It is useful in the design of experiments or algorithm development to quantify the effect of changing system parameters across all user performance levels. For example, the effect of the data collection limit on the amount of data required to achieve a certain accuracy given a user's performance level. Unsurprisingly, the data collection limit has little effect at very low and very high detectability indices. Users with low performance levels usually have minimally distinct classifier likelihoods (low d) and represent those who cannot use the BCI system efficiently due to the large amount

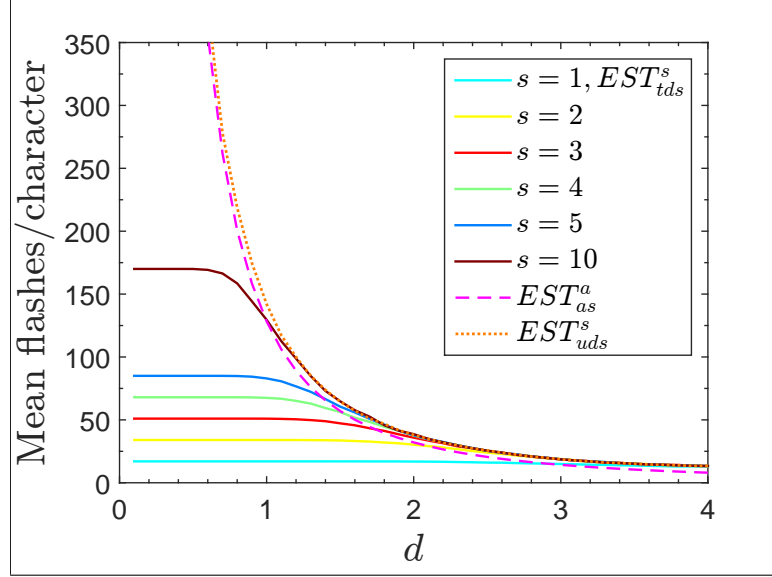


FIGURE 3.4: Expected stopping time, EST, vs. detectability index, d , with the Bayesian dynamic stopping (DS) algorithm for a row-column paradigm with a 9×8 grid. P300 spelling runs with the Bayesian DS algorithm were simulated using normal distributions (for target and non-target likelihoods) and a sequence limit, s . The maximum number of flashes, $t_{\max} = s \times (9 + 8)$ flashes. EST_{as}^a = asymptotic lower bound for the untruncated Bayesian DS algorithm with probability threshold, $P_{th} = 0.9$, determined analytically (Algorithm 2); and EST_{uds}^s = untruncated Bayesian DS with $P_{th} = 0.9$, determined via simulations; EST_{tds}^s = truncated Bayesian DS with $P_{th} = 0.9$, and data collection limit t_{\max} , determined via simulations.

of data collection needed to achieve acceptable accuracy levels for communication. Users with high performance levels (high d) require only a small amount of data to use the BCI system effectively. The impact of the data collection limit on accuracy is most noticeable at the mid-range detectability indices. Hence, for users with mid-range performance levels, an appropriate data collection limit can significantly increase their maximum attainable accuracy.

Figure 3.4 shows the corresponding expected stopping time with the untruncated and truncated Bayesian DS algorithms for the RCP, to verify the expected stopping time determined analytically, EST_{as}^a , (Algorithm 2). The expected stopping time of the untruncated DS algorithm, EST_{uds}^s , is asymptotically lower- bounded by EST_{as}^a , as $EST_{as}^a < EST_{uds}^s$. The expected stopping time of the truncated DS algorithm,

EST_{ids}^a , more or less follows EST_{uds}^s , but starts to deviate as it approaches and converges with the data collection limit. These results indicate that EST_{as}^a provides an analytic expression for the lower bound of the EST of the untruncated DS algorithm, from which the EST for the truncated DS algorithm can be inferred. A better estimate for the EST with the truncated Bayesian DS algorithm can be obtained empirically via MC simulations.

Some stimulus paradigms in the BCI literature do not satisfy the properties necessary to use the analytical solutions for the performance estimates for the RCP. However, for generic stimulus paradigms for which property 3 is satisfied, an alternative method for estimating their accuracy was proposed (Algorithm 3). For validation, other stimulus paradigms from the literature were assessed. The checkerboard paradigm (CBP) [50] differs from RCP in that the set of flash groups are created from a virtual overlay of a checkerboard pattern on the grid do not satisfy property 1 and 2. Character selection for the CBP cannot be partitioned into a two-stage selection process and the spatial restrictions of the CBP may lead to flash groups of unequal sizes. In addition, it does not satisfy property 3 as there is character-to-flash group randomisation after each sequence. However, if it is assumed that there is no character-to-flash group randomisation, Algorithm 3 can be used to approximate a lower bound of the accuracy for the CBP as implemented in [50].

The projected accuracy, determined analytically using Algorithm 3, A_{fss}^a , is compared to the accuracy determined via simulations of the Bayesian FSS algorithm for both the RCP and CBP. Both paradigms were implemented with a 9×8 grid. It should be noted that for the CBP as implemented in Townsend *et al.*, (2010) [50], there is character-to-flash group randomisation after each sequence. However, for this application, it is assumed that the base stimulus pattern for the CBP is replicated based on the number of sequences, with no character-to-flash group randomisation. This is required for calculating the projected accuracy analytically, and therefore nec-

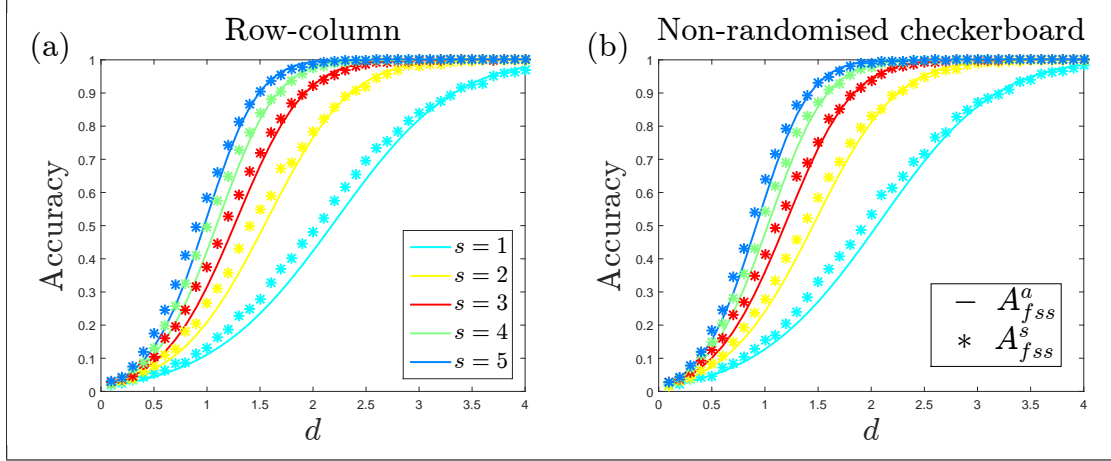


FIGURE 3.5: Accuracy, A , vs. detectability index, d , with the Bayesian algorithm with fixed sample size (FSS) for (a) a row-column paradigm (RCP) and, (b) a non-randomised checkerboard paradigm (CBP), both implemented with a 9×8 grid. P300 spelling runs with the Bayesian FSS algorithm were simulated using normal distributions (for target and non-target likelihoods) and a sequence limit, s . The maximum number of flashes is, $t_{\max} = s \times (9 + 8)$ flashes for the RCP and $t_{\max} = s \times 24$ flashes for the CBP. $A_{f_{ss}}^a$ = projected accuracy with the Bayesian FSS with data collection limit t_{\max} , determined analytically (Algorithm 3); and $A_{f_{ss}}^s$ = projected accuracy with Bayesian FSS with data collection limit t_{\max} , determined via simulations.

essary to compare the analytical- and simulated-based outcomes, thereby verifying the methodology proposed in Algorithm 3.

Figure 3.5 compares the analytical and simulation results for the RCP and a non-randomised CBP. For both paradigms, it can be observed that $A_{f_{ss}}^s$ closely follows $A_{f_{ss}}^a$, with some noticeable amount of deviation, although the difference reduces with an increased number of sequences. These errors arise as a result of approximating the pdf of the maximum of correlated random variables with a normal distribution [105], whereas the actual pdf might be skewed [106]. For example, for the RCP, the approximation error for the Bayesian FSS algorithm is higher with Algorithm 3 ($\text{MSE} = 7.1 \times 10^{-2}$), compared to Algorithm 1 ($\text{MSE} = 2.1 \times 10^{-5}$) where the cdf of the maximum of independent random variables is defined exactly. Thus, the match between simulations and analytical estimation for the RCP is less accurate

in Figure 3.5(a) than in Figure 3.3(a). Although the deviation between simulations and analytical estimation is larger for Algorithm 3, the differences are small for likely data collection limits (i.e. number of sequences, $s > 1$). Thus, these results provide verification for the methodology proposed in Algorithm 3 for generic stimulus paradigms that do not satisfy the properties of the RCP.

Overall, these results indicate that the projected performance for the truncated Bayesian DS algorithm can be approximated analytically from the equivalent likelihood ratio formulation. For accuracy, some amount of error may result from approximating the truncated DS algorithm with a FSS algorithm, or approximating the probability distribution of the maximum of correlated random variables. From the asymptotic lower bound of the untruncated DS algorithm, the expected stopping time of the truncated DS time can be inferred, after considering deviations due to data truncation. Alternatively, the projected performance measures of the truncated Bayesian DS can be determined empirically via MC simulations of P300 spelling runs. Under conditions of normality, the detectability index can be used to parameterise the respective performance measures, whether determined analytically or by simulations.

3.3.2 Analyses with BCI Datasets

The previous experiments demonstrated that the detectability index can be used to estimate P300 speller performance under a normality assumption. The proposed methods to approximate system performance are now validated using data from on-line BCI studies conducted using the truncated Bayesian DS algorithm. For the original experiments, the class conditional pdfs used in the DS algorithm implemented for a given user, as described in Throckmorton *et al.*, (2013) [25], were obtained via kernel density estimation (KDE) rather than estimated to be Gaussian. For non-parametric distributions such as those that would result from KDE, ana-

lytically determining the proposed performance measures would involve numerical recursive convolutions and integration of pdfs and cdfs. The projected performances can still be obtained via MC simulations. However, without the normality assumption, performance cannot be parameterised by a single value, making it inconvenient for performance visualisation and analysis.

However, this issue might be addressed if the differences between simulated performance estimates using non-parametric distributions and analytical performance estimates under a normality assumption are negligible, as might be the case if the non-parametric distributions closely resemble normal distributions. Work by Colwell *et al.*, (2014) supports the assumption that the class conditional pdfs may be assumed to be normal distributions [53]. In the analysis in Colwell *et al.*, the accuracy of P300 speller simulations using classifier scores drawn from bootstrapped user data (equivalent to non-parametric estimation of the pdfs) was observed to converge to the projected accuracy estimated using normal pdfs, suggesting that the pdfs closely model normal distributions. Since the finite space pdfs obtained via KDE closely model normal distributions, it is hypothesised that the detectability index can still be used to closely approximate the projected performance measures. To verify this assumption, data obtained from previous online BCI studies where the Bayesian DS algorithm was implemented with non-parameteric pdfs are analysed and compared to simulations using normal distributions.

(i) Simulations with EEG Data

Previously, the analytical methods of estimating projected accuracy and expected stopping time were validated against simulations that relied on normal distributions for the class conditional classifier likelihood pdfs. These performance estimates derived from simulations with synthetic data under the normality assumption are referred to as the *predicted* accuracy, A_{pr} , and expected stopping time, EST_{pr} .

In these offline simulations, validation is provided by comparing the predicted performances to those from simulations based on participant data collected in the Throckmorton *et al.*, (2013) study [25]. First, the classifier pdfs for the Bayesian DS algorithm, i.e. $p(y|H_0)$ and $p(y|H_1)$, to be used in the probability update process, were developed via KDE from the training data. For the simulations, P300 spelling runs with 9×8 RCP were simulated using bootstrapped classifier scores obtained from the EEG data collected online during the testing run. The probability threshold was set at $P_{th} = 0.9$, and the number of sequences varied from 1 to 5. The results from the simulations using participant EEG data are referred to as the *observed* performances: A_{obs} and EST_{obs} . For comparison to the analytical estimates, the detectability indices of the training and test data were computed from the respective class conditional classifier scores.

Figure 3.6(a) shows a comparison of the observed and the predicted performances using detectability indices computed from the training data, as would be expected during real-world BCI use. Even though EST_{obs} closely follows EST_{pr} , it can be observed across participants that A_{pr} generally tends to overestimate A_{obs} . Some of this mismatch may be due to a drift in the class conditional probability distributions from training to testing data. As a second comparison, the test data was used to determine the detectability indices, providing a closer match between detectability index and the data used to estimate performance. As can be seen in Figure 3.6(b), the trend of the observed performances is closer to those predicted by d of the test data.

The results indicate that the detectability index can be used to predict user performance, confirming that the pdfs obtained via KDE closely approximate normal distributions. These results also highlight the limitations of using training data to predict BCI performance; d derived from the test data provided a better estimate of the observed performances than d derived from the training data. However, the

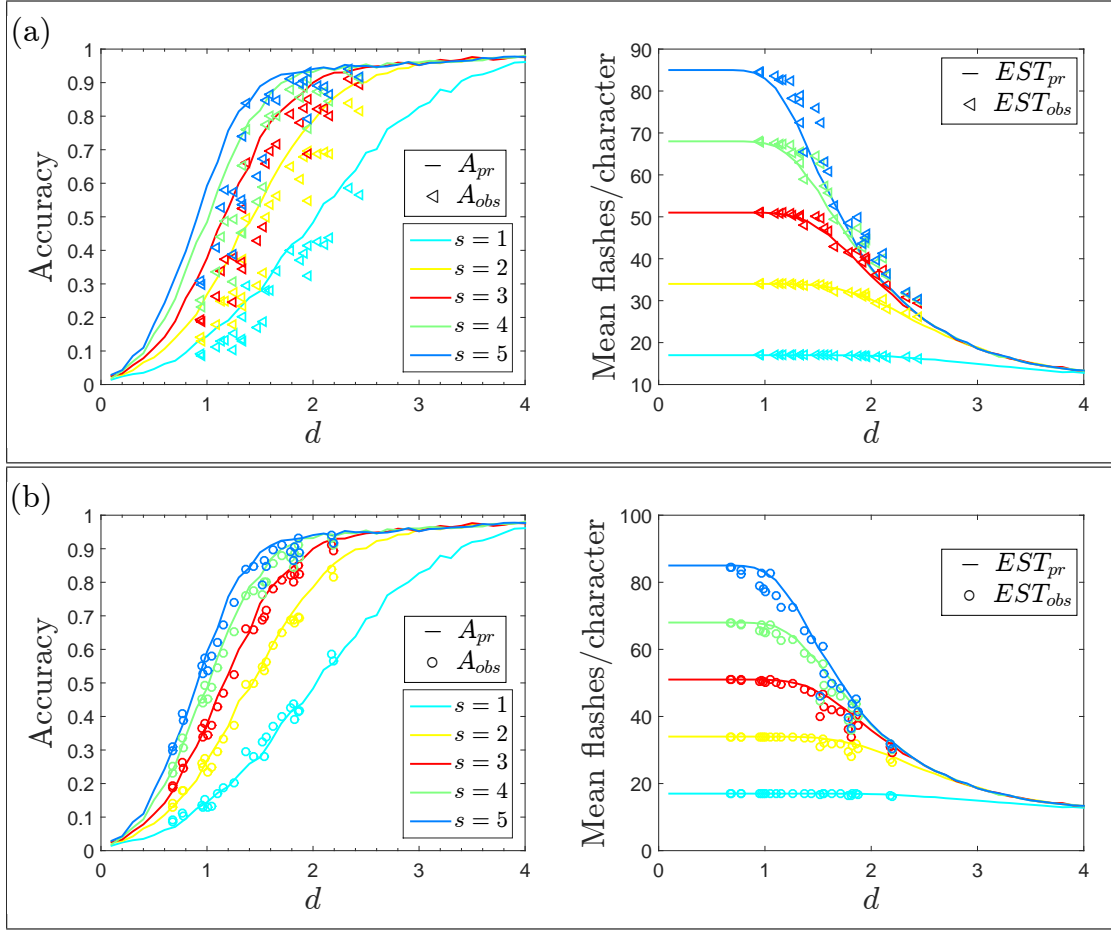


FIGURE 3.6: Performance vs. detectability index, d , for the 9×8 row-column paradigm, obtained from simulating Bayesian dynamic stopping (DS) with participant EEG data. For each participant, the classifier weight vector and likelihood functions (via kernel density estimation) for the DS algorithm were developed from the training data. P300 spelling runs were simulated using the test data with a probability threshold, $P_{th} = 0.9$, and sequence limit, s . The maximum number of flashes, $t_{\max} = s \times (9 + 8)$ flashes. The predicted performances, accuracy, A_{pr} and expected stopping time, EST_{pr} , were obtained from another set of P300 speller simulations with synthetic data using normal distributions. (a) Comparison of the observed performances obtained from the simulations with participant data, A_{obs} and EST_{obs} , to that predicted by d of the training data. (b) Comparison of the observed performances to that predicted by d of the test data.

training data can provide useful prior information about a user’s performance level and serve as a guide for setting the amount of data collection needed to achieve a desired accuracy level.

(ii) *Comparison to Online Data*

In the previous section, the observed accuracies and expected stopping times were computed from simulations that relied on participant EEG data. In this section, reported performances from previous online studies where the Bayesian DS algorithm ($P_{th} = 0.9$) was implemented are compared to predicted performances for further validation. In Throckmorton *et al.*, (2013) [25] and Mainsah *et al.*, (2014a) [59], non-disabled participants performed word copy-spelling tasks of 30 and 36 characters, respectively, using the RCP on a 9×8 grid, with a limit of 10 sequences. In Mainsah *et al.*, (2015) [88], participants with ALS performed word copy-spelling tasks of 12 characters¹ using the CBP on a 6×6 grid, with a limit of 7 sequences.

The predicted performances for the Bayesian DS algorithm were generated using simulations with normal distributions according to the respective algorithm parameters, i.e. stimulus paradigm, sequence limit and stopping thresholds. Since these data are intended for validation, the detectability index, d , for each participant session was computed from the class conditional classifier scores of the test dataset and used to predict performance.

Figure 3.7 shows the results for both non-disabled participant studies, and Figure 3.8 shows the results for the ALS participant study. Since the online estimation relies on a small number of samples, unlike the bootstrapping simulations from the previous section, the online performance estimates have low resolution and may exhibit higher variation. Nonetheless, it can be observed qualitatively that the trends of the observed performances follow those predicted by the user’s detectability index,

¹The participants underwent three P300 speller sessions, with 12 characters copy-spelled per session. The results from each session were analysed separately.

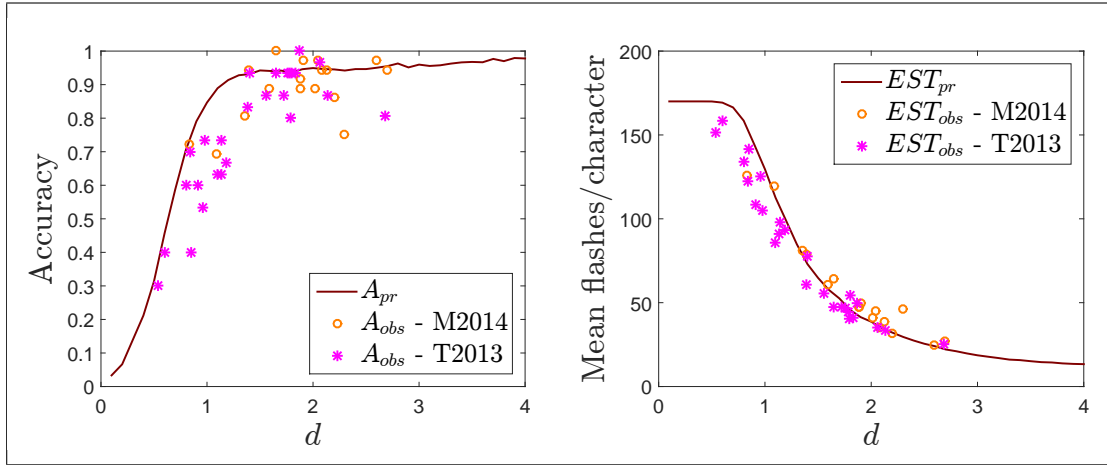


FIGURE 3.7: Performance vs. detectability index, d , obtained from online studies using Bayesian dynamic stopping (DS) with a 9×8 row-column paradigm, a probability threshold, $P_{th} = 0.9$ and a sequence limit of 10 (17 flashes/sequence). The predicted accuracy, A_{pr} , and expected stopping time, EST_{pr} , were obtained from P300 speller simulations with normal distributions. The observed performances, A_{obs} and EST_{pr} , were estimated from copy-spelling 30 and 36 characters in Throckmorton *et al.* (T2013) [25] and Mainsah *et al.* (M2014) [59], respectively. The left plot compares A_{pr} and A_{obs} . The right plot compares EST_{pr} and EST_{obs} .

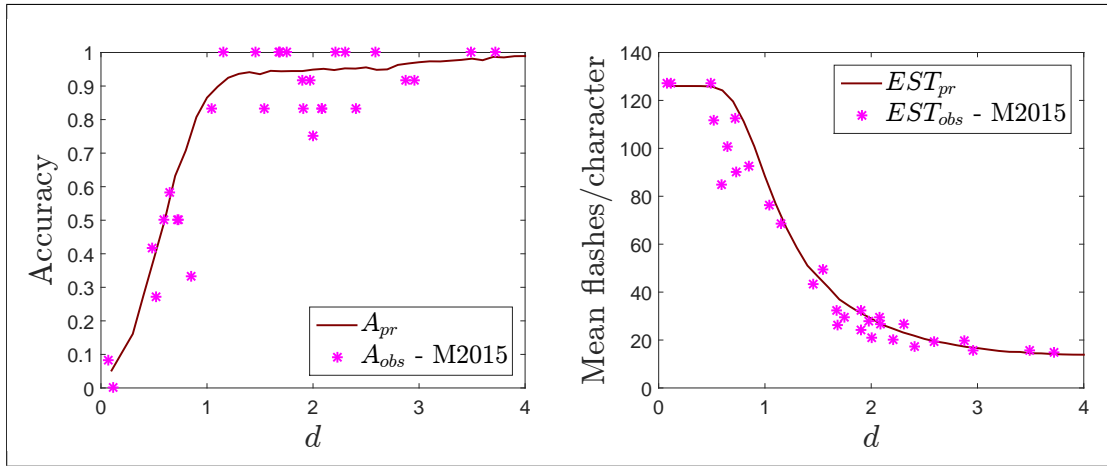


FIGURE 3.8: Performance vs. detectability index, d , obtained from online studies using Bayesian dynamic stopping (DS) with a 6×6 checkerboard paradigm, a probability threshold, $P_{th} = 0.9$ and a sequence limit of 7 (18 flashes/sequence). The predicted accuracy, A_{pr} , and expected stopping time, EST_{pr} , were obtained from P300 speller simulations with normal distributions. The observed performances, A_{obs} and EST_{obs} , were estimated from copy-spelling 12 characters in Mainsah *et al.* (M2015) [88]. The left plot compares A_{pr} and A_{obs} . The right plot compares EST_{pr} and EST_{obs} .

for the two stimulus paradigms at the respective sequence limits. These results are a final confirmation that the detectability index can provide a useful means to assess BCI performance without the need for extensive online testing.

3.4 Discussion

Previous approaches to predict BCI performance relied on developing features that correlated with user performance and obtaining empirical estimates from user data, making their estimates easily susceptible to user variation. A model-based approach is preferable as it is robust to noisy estimates. In this work, a theoretical model is proposed for the P300 speller with the Bayesian algorithm that provides a principled method to estimate a user's performance. Although the focus of this work was on P300 spellers, this model-based approach could be applied to any decision-based BCI in which 1-of- M choices is selected.

Online BCI performance evaluation usually relies on a small number of samples, resulting in estimates with high variance and low resolution. The detectability index of a user can be used to predict long-term performance and is easily computed from parameters of the class conditional classifier scores of a given dataset. However, predictions based on a given dataset may not necessarily be applicable to subsequent data collections depending on the significance of the shift in the data statistics. From our analysis, the predicted performances based on the training data tended to overestimate the observed performances, likely due to a model mismatch. This may not be surprising given the non-stationarity of EEG data or other factors such as increasing fatigue with system use. However, the training data provides useful prior information to ballpark a user's performance level. With the detectability index, possible changes in user performance level from calibration can be accounted for during analysis. This information when coupled with the knowledge that the performance curves obtained from the training data may be an upper bound, can

provide guidance for user parameter selection, e.g. a data collection limit.

The ability to predict user performance is also an important tool to pre-assess BCI prior to online use. Due to the time, expense and resources required for online BCI implementation, offline BCI algorithm development and analysis is necessary to identify and study promising alternatives prior to online testing. The ability to quantify user performance via the detectability index enables the impact of a new algorithm or modification, parameter value, strategy, etc., to easily be visualised across all performance levels. For example, the analysis in this study revealed for users with mid-range performance levels, the data collection limit had a noticeable impact on their accuracy level, whereas the impact was negligible in users with very high and very low performance levels.

In this chapter, a new method was developed which uses the classifier detectability index to predict performance of the Bayesian DS algorithm in the P300 speller. The P300 speller is simplified from an M -ary to a binary hypothesis problem, by exploiting the equivalent likelihood ratio formulation of the Bayesian probability space. The performance measures can be obtained either analytically or via Monte Carlo simulations, and this approach is applicable across stimulus paradigms. The proposed method was verified with data from previous online studies, using participants with and without disability, different stimulus paradigms and data collection limits. The observed online performances followed the trends predicted by the detectability index of the respective test dataset. Overall, an important tool is provided for offline algorithm development and analysis of BCI performance across a range of user performance levels without extensive online testing.

Controlled Stimulus Selection

Evidence from the literature suggests that changing stimulus paradigm parameters, such as the timing of stimulus presentation, the grid spatial layout, and the order of presentation of non-target/target stimulus events, can affect BCI performance [56, 57, 58]. Some studies have focused mainly on changing cosmetic aspects of the visual paradigm to enhance user focus or minimise user distractions [50, 62, 63, 64, 65]. However, with these approaches, there is random generation and presentation of flash groups without consideration of maximising the information content of what is presented to the user.

A potential target character is encoded by its flash pattern which can be considered as repetitions of a binary codeword, where each bit represents its presence or absence in a flash group. Using principles from coding theory [22, 23], the codeword for each character can be chosen such that errors are minimised in the decoding process. Designing a stimulus paradigm with an information-theoretic approach provides a more principled way for determining the composition of flash groups and their presentation order to enhance performance. Some previous studies have exploited coding theory in designing stimulus paradigms [70, 71, 72, 107]. These previously

designed stimulus paradigms designed based on coding theory showed the potential to reduce errors and improve performance over the RCP, based on either theoretical analysis or offline simulations to estimate performance. However, these paradigms ultimately performed similarly to or worse than the RCP when tested online due to a lack of consideration of the physiology of P300 signal elicitation, such as target-to-target (TTI)-related effects, when predicting online performance.

This chapter proposes a new method that considers the competing interests of information theoretics and physiological effects to develop a new stimulus paradigm that improves performance relative to the RCP, which continues to be the most common stimulus presentation paradigm used among P300-based spellers. A method to predict BCI performance for a generic stimulus paradigm based on the detectability index was proposed and validated with online studies in chapter 3. This performance prediction method is used to develop a new stimulus paradigm through an iterative search of codewords that maximise the distinction between character cumulative responses while taking into account physiological limitations due to refractory effects. This new stimulus paradigm is referred to as the *performance-based paradigm*.

4.1 The Performance-based Paradigm

The process of developing a performance-based paradigm (PBP) involves the selection, from an exponentially large search space, of a subset of codewords that maximises the performance of the character cumulative response function, $\Theta_m(t)$. For example, for an (M, l) -code with a binary alphabet, this requires selecting M codewords of length l out of 2^l possibilities. Within the context of the Bayesian DS algorithm, the objective is selecting an (M, l) -code, with repetition, that minimises the expected stopping time (EST) subject to accuracy levels above or equal to the probability threshold, i.e. $A \geq P_{th}$. For a stimulus paradigm, the EST of a truncated Bayesian DS algorithm, i.e. with an imposed data collection limit, follows that of the

untruncated DS algorithm until it approaches and converges with the data collection limit (Figure 3.4). An equivalent result is achieved by minimising the EST for one sequence or codebook instantiation of the stimulus paradigm.

In searching for an improved codebook, the performances of the codebooks are compared using simulations to select the best-performing codebook for online testing. However, a mismatch can occur between offline predictions and online results. In the literature, several new stimulus paradigms were developed and hypothesised to perform better than the 6×6 RCP: RIPRand (1/6) [104], D8 and D10 [71]; however, online results demonstrated similar or decreased performance. The RIPRand (1/6) paradigm [104] has the same codebook as the 6×6 RCP, but with character-to-flash group randomisation after each sequence rather than flash-group order randomisation. Character-to-flash group randomisation is hypothesised to sometimes improve performance by minimising the temporal correlation between the character cumulative responses. The D10 paradigm maximises Hamming distance separation between codewords with a minimum Hamming distance, $d_{\min}^H = 10$, imposed. The D8 paradigm minimises pairwise codeword confusion and overall decoding errors and has a $d_{\min}^H = 8$. In general, the higher the d_{\min}^H of a codebook, the better its error-correcting capability (2.5).

Figure 4.1 shows results of simulating P300 spelling runs with participant EEG data (Algorithm 4) with the three previously mentioned stimulus paradigms, as well as the RCP. Each participant represents a performance level quantified by the detectability index. The results from the simulations indicate improved accuracy with reduced EST for the three stimulus paradigms when compared to the RCP. However, the results from online studies indicate the three stimulus paradigms performed similarly or worse than the RCP [71, 104]. It is hypothesised that TTI-related effects contributed to the significantly worse performance of the D10 paradigm compared to RCP. Although the D10 paradigm ($d_{\min}^H = 10$) possesses better Hamming properties

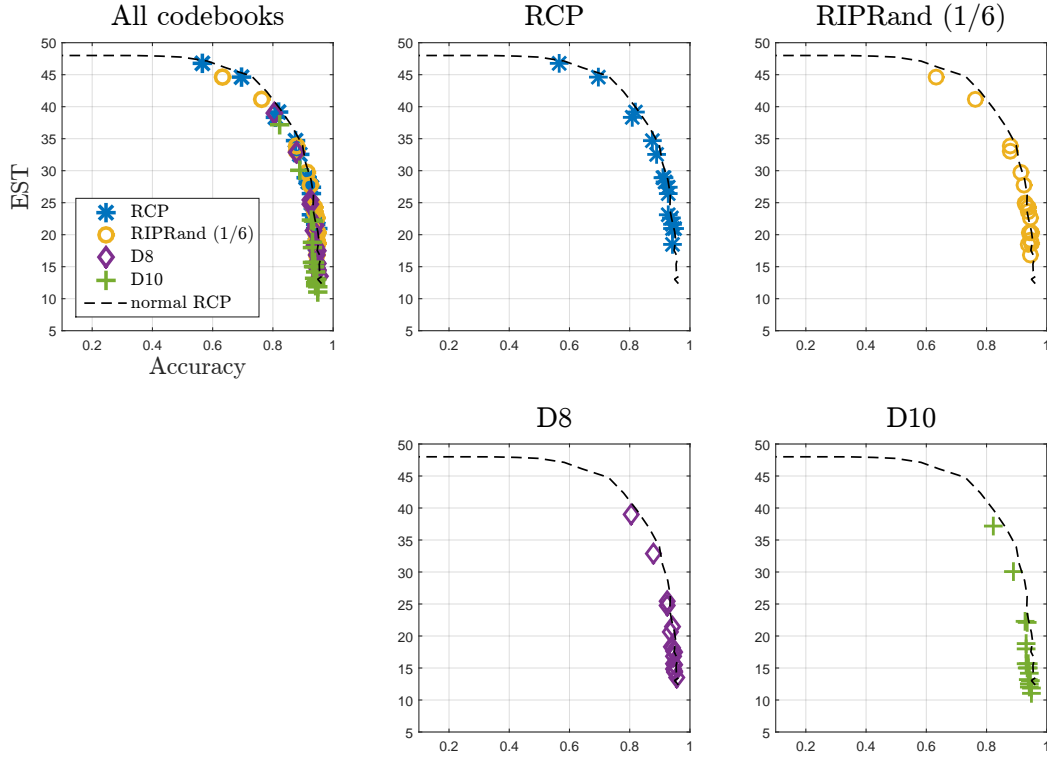


FIGURE 4.1: Performance comparison for different stimulus paradigms with $(36,24)$ codes: expected stopping time (EST) vs. accuracy. An (M, l) -code is a codebook with M codewords of length l . The stimulus paradigms include: the conventional row-column paradigm (RCP); the restricted isometric property paradigm (RIPRand) with a $1/6$ flash rate, which has the same codebook as the RCP but with character-to-codeword randomisation after each sequence [104]; and the minimum Hamming distance paradigms, D8 and D10 [71]. The $(36,12)$ -codes, RCP and RIPRand ($1/6$), are doubled to create $(36,24)$ codes. Using participant EEG data from [59], P300 spelling runs were simulated using the Bayesian dynamic stopping algorithm with probability threshold, $P_{th} = 0.9$ and data collection limit, $t_{max} = 48$. Another set of simulations was performed with normal distributions over a range of detectability indices using the RCP.

than the RCP ($d_{min}^H = 4$), it is characterised by a relatively large proportion of short TTIs, $\approx 40\%$ TTI 1 and 25% TTI 2, compared to RCP, $\approx 11\%$ TTI 1 and 11% TTI 2, based on TTI analysis from the simulations.

Although offline simulations serve as a convenient means to quickly identify and test a few promising alternatives prior online testing and validation, it is important

that the correct assumptions are made prior to interpreting results from simulations. The proposed method to predict performance presented in chapter 3 compares the performance across stimulus paradigms given the same detectability index value. In the previous simulations, it was assumed that the detectability index does not change across the different stimulus paradigms. However, for a user, the detectability index can significantly change due to the stimulus paradigm parameters; specifically, negative TTI-related effects due to short TTIs can lead to poorer performance. A better approach to compare stimulus paradigms is to predict how detectability index changes based on the stimulus paradigm parameters prior to comparing their performance via simulations. Such a method requires modeling the P300 speller as a channel with memory.

4.1.1 P300 Channel with Memory

Unlike the previous codebook-based paradigm approaches, this work assumes a P300 channel with memory due to TTI-related effects. For example, in a P300 speller channel with a TTI memory factor of 1, the conditional probability of a classifier score based on the previous target stimulus events is:

$$p(y_j|x_i, x_j), \quad i < j, \quad x_i = x_j = 1 \quad (4.1)$$

where y_j is the current classifier score; x_i and x_j are the current and previous target character presentations, respectively; and $p(y_j|x_i, x_j)$ is the conditional probability of observing the classifier score, y_j , given the previous and current target stimulus event, x_i and x_j , respectively. It is hypothesised that including TTI-related effects in a P300 speller simulation model would better predict performance changes across different stimulus paradigms.

In this work, an attempt was made to incorporate TTI-related dependencies, such as in (4.1), into a graphical model representation of the P300 speller and using this

model for simulations. The analysis relied on using TTI-based features from participant EEG data using the RCP to construct a model of a new stimulus paradigm given its TTI statistics. However, this approach failed to result in a good physiological model to predict changes in a user's detectability index due to TTI-related effects. A similar approach was used in [70, 71] to account for TTI-related effects in the classifier development process as well as the decoding algorithm. However, this did not compensate for the negative TTI-related effects in their proposed stimulus paradigms which performed similarly or worse than the RCP. It is hypothesised that the TTI-related effects due to the occurrence of double target character stimulus events interspersed among non-target stimulus events (e.g. ...00110000001100...) might be less detrimental to performance compared to continuous target character presentation (e.g. ...001111111100...). The former TTI scenario is more likely to occur in the RCP, while the latter is predominant in the stimulus paradigms proposed in [70, 71, 72].

An alternative approach is proposed here to develop a new stimulus paradigm that maintains similar detectability index in a user compared to the RCP, but still improves performance based on the stimulus paradigm parameters. The large 2^l search space is narrowed down to a smaller search space based on specific parameters that minimise TTI-related effects and positively affect performance such that offline predictions may better reflect online performance. A minimisation problem to select a codebook is defined, with the EST as the objective function and constraints placed

on the codewords in the new codebook:

$$\min_{\mathbf{C}^{M \times l}} \text{EST}(\mathbf{C}^{M \times l}, d) \quad (4.2a)$$

$$\text{subject to } d_{\min}^H(\mathbf{C}) \geq D \quad (4.2b)$$

$$|i - j| \geq \text{TTI}_{\min}, \forall c_{m,i} = c_{m,j} = 1 \quad (4.2c)$$

$$\omega(\mathbf{c}_m) \in \omega \quad (4.2d)$$

where $\mathbf{C}^{M \times l} = \{\mathbf{c}_m\}_{m=1}^M$ is an (M, l) -code, consisting of l -bit codewords, $\mathbf{c}_m = [c_{m,1}, c_{m,2}, \dots, c_{m,l}]$; and (4.2b-4.2d) represent the performance-based parameter constraints. These parameters include the minimum Hamming distance, d_{\min}^H , the minimum target-to-target interval, TTI_{\min} , and the target presentation frequency in a codeword, $\omega(\mathbf{c}_m)$. The selection of these parameters is discussed in the next section.

4.1.2 Performance-based Parameters

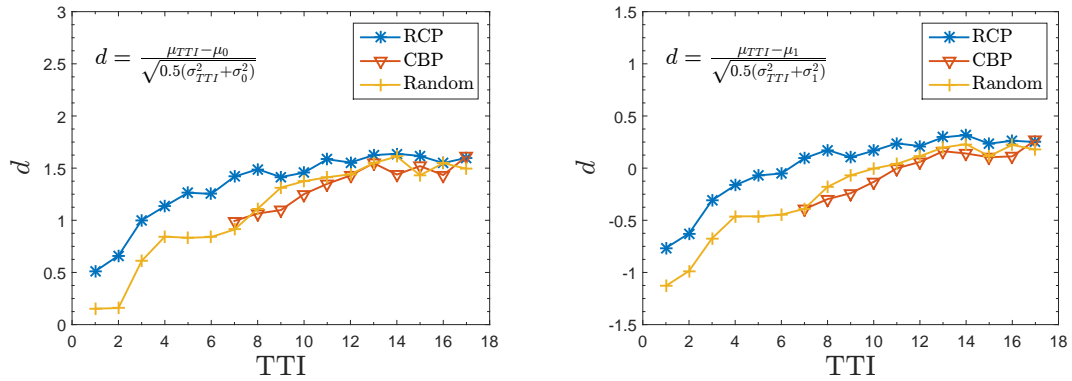
A stimulus paradigm is initially defined by its codebook of size, $|\mathbf{C}| = M$, and codeword length, l . The performance-based parameters are used to narrow down the 2^l search space to a smaller space from which to select M codewords. The higher the d_{\min}^H of a codebook, the greater the error-correcting capability in the decoding process (2.5): in theory, a codebook with d_{\min}^H greater than that of the RCP will have better error-correcting capability.

The minimum target-to-target interval, TTI_{\min} , and the target presentation frequency, $\omega(\mathbf{c}_m)$, in a codeword are interrelated. In general, the more a target character is presented, the greater the possibility of generating a high classifier score drawn from the target classifier score pdf. However, for a fixed codeword length, the interval between target character presentations or the TTI is affected by target presentation frequency. Higher target presentation frequencies result in shorter TTIs which can be detrimental to performance due to the higher possibility of generating classifier

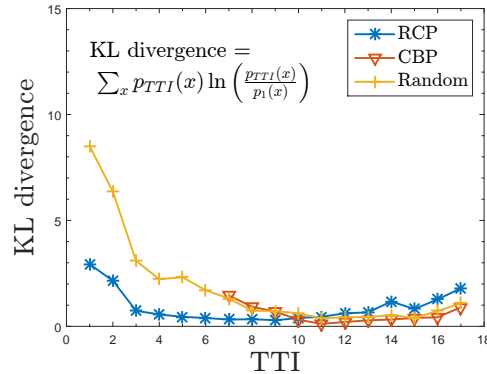
scores that are closer to the non-target classifier score pdf (Figure 2.5).

Selecting a low TTI_{\min} value that minimises TTI-related effects while also maximising target classifier scores depends on the presentation frequency of the stimulus paradigm as determined by the flash duration and inter-stimulus interval (ISI). While TTI-related effects can be minimised by using low presentation frequencies or imposing a high TTI_{\min} (e.g. [50]), this can reduce spelling speed or increase the EST in the case of a dynamic stopping algorithm. To determine an appropriate TTI_{\min} , participant EEG data were analysed to compare the discriminability of TTI-segregated classifier scores to that of the aggregate non-target and target classifier scores for three stimulus presentation paradigms: RCP, checkerboard (CBP) and random paradigms. In the random paradigm, the characters in the flash groups are randomly generated, with the condition that each character is flashed once before any other character is flashed again. The CBP is a special case of the random paradigm where a minimum TTI is imposed and spatial restrictions, with respect to the grid layout, are placed on the composition of characters in a flash group.

For this dataset, in all three paradigms, both the flash duration and ISI were set to 62.5 ms. Due to a low frequency of occurrence of some TTIs in certain stimulus paradigms, for each participant, data were pooled across calibration and test runs and 10-fold cross validation was performed to train and test classifiers. The resulting classifier scores were aggregated by target and non-target classifier scores. In addition, the target classifier scores were segregated by TTI. For each set of grouped scores, the mean and the variance were determined, as well as the likelihood pdf generated via KDE from score histograms, as in Figure 2.5. Figure 4.2(a)-(b) shows the detectability indices between TTI-segregated classifier scores and non-target or target classifier scores, respectively, averaged across participants. The results confirm the trend that for shorter TTIs, the TTI-segregated pdf is closer to the aggregate non-target pdf. As the TTI increases, the target TTI-segregated



(a) Detectability index, d , between TTI-segregated and non-target classifier scores (b) Detectability index, d , between TTI-segregated and target classifier scores



(c) Kullback-Leibler (KL) divergence between TTI-segregated and target classifier score pdfs

FIGURE 4.2: Comparison of the discriminability between target-to-target interval (TTI)-segregated classifier scores ($p_{TTI}, \mu_{TTI}, \sigma_{TTI}^2$), and the aggregate non-target (p_0, μ_0, σ_0^2) or target classifier scores (p_1, μ_1, σ_1^2) of three stimulus paradigms: row-column (RCP), checkerboard (CBP) [50], and random paradigm. (p_i, μ_i, σ_i^2) denotes the pdf (p_i), mean (μ_i) and variance (σ_i^2), of classifier scores grouped by a condition denoted by i . The flash duration and inter-stimulus interval were both set to 62.5 ms.

pdf approaches the aggregate target pdf.

Figure 4.2(c) shows the Kullback-Leibler (KL) divergence [108] between the TTI-segregated and aggregate target classifier score pdfs. The KL divergence is another metric that quantifies the distance between two pdfs and provides a better visual measure of the degree of overlap of two pdfs. For example, for the participant result

shown in Figure 2.5, from TTI 3, the TTI-segregated pdf more or less completely overlaps with the aggregate target pdf. From Figure 4.2(c), the TTI-segregated pdfs of shorter TTIs for the random paradigm are noticeably dissimilar from the aggregate target pdf. These TTI-related effects are minimised in the CBP where a TTI_{\min} of 7 is imposed. For the RCP, the pdfs for short TTIs (TTI 1 and 2) are most dissimilar to the target pdf, although to a lesser degree than the random paradigm. From TTI 3, the target TTI-specific pdf more or less converges to that of the target pdf. Based on these results, for both an ISI and flash duration of 62.5ms, a suitable TTI_{\min} is a TTI of 3, which corresponds to a time interval of 375 ms between target character presentations.

4.1.3 Codebook Development Algorithm

The approach used in this work towards solving the combinatorial problem defined in (4.2) is to build a codebook using an iterative greedy search where a new codeword is added to a partially-filled codebook such that the objective function is minimised with respect to the other codewords. However, the objective function, the expected stopping time (4.2a), to be minimised over the set of codewords does not have a tractable analytical solution for a generic stimulus paradigm, except for the RCP (Algorithm 2).

Alternatively, an analytical solution to approximate the accuracy, A_{pr} , with a generic stimulus paradigm was proposed and validated in chapter 3 (Algorithm 3). From Figure 4.1, it can be observed that for an (M, l) -code, the relationship between A_{pr} and EST is defined along a fixed curve. Based on the offline simulations, the performance improves from $\text{RCP} < \text{RIPRand} (1/6) < \text{D8} < \text{D10}$. An improved accuracy is observed due to the better Hamming distance properties from RCP to D10. The EST is reduced because of the shorter TTIs from RCP to D10 paradigm, especially due to a higher target presentation frequencies in the D8 and D10 paradigms.

Algorithm 5: Performance-based Codebook Development

Input: M = number of codewords in codebook; l = codeword length; ω = codeword weight; d_{\min}^H = minimum Hamming distance; TTI_{\min} = minimum target-to-target interval

Output: \mathcal{C}_{new} = New codebook

- 1 **Function** searchCodewords($M, l, \omega, d_{\min}, \text{TTI}_{\min}$)
- 2 $\triangleright \mathcal{X} \stackrel{\text{add}}{\leftarrow} \mathbf{x}$: add codeword \mathbf{x} to codebook \mathcal{X}
- 3 $\triangleright \mathcal{X} \stackrel{\text{remove}}{\rightarrow} \mathbf{x}$: remove codeword \mathbf{x} from codebook \mathcal{X}
- 4 - Narrow down 2^l codeword space by eliminating codewords with proportion of 1's, $w \notin \omega$, $\text{TTI}(\mathbf{x}) < \text{TTI}_{\min}$ and pairwise Hamming distance, $d^H(\mathbf{x}, \mathbf{y}) < d_{\min}^H$
- 5 - Current codeword search space, \mathcal{C}_{old}
- 6 - Initialise \mathcal{C}_{new} with codeword with maximum $d^H(\mathbf{x}, \mathbf{y})$ with respect to other codewords in \mathcal{C}_{old} .
- 7 - $\mathcal{C}_{\text{old}} \stackrel{\text{remove}}{\rightarrow} \mathcal{C}_{\text{new}}$
- 8 **while** $|\mathcal{C}_{\text{new}}| < M$ **do**
- 9 **for** $i = 1 : |\mathcal{C}_{\text{old}}|$ **do**
- 10 - $c_i = \mathcal{C}_{\text{old}}(i)$
- 11 - $\mathcal{C}_{\text{temp}} = \mathcal{C}_{\text{new}} \stackrel{\text{add}}{\leftarrow} c_i$
- 12 - $\text{tempAcc}(i) = \text{paradigmAccuracy}(d, \mathcal{C}_{\text{temp}}, 1, \{P_i\}_{i=1}^{|\mathcal{C}_{\text{temp}}|})$ \triangleright (Algorithm 3)
- 13 **end**
- 14 - $c_{\text{new}} = \max_i \{\text{tempAcc}(i)\}_{i=1}^{|\mathcal{C}_{\text{old}}|}$
- 15 - $\mathcal{C}_{\text{new}} \stackrel{\text{add}}{\leftarrow} c_{\text{new}}$
- 16 - $\mathcal{C}_{\text{old}} \stackrel{\text{remove}}{\rightarrow} c_{\text{new}}$
- 17 **end**
- 18 **return** \mathcal{C}_{new}

Hence, for an (M, l) code, an inverse relationship exists such that if A_{pr} is maximised, inherently, EST is minimised.

Algorithm 5 outlines pseudo-code to develop a new codebook based on preset performance-based parameters by maximising A_{pr} . The proposed method to develop a performance-based paradigm (PBP) was used to generate a new codebook using the 6×6 RCP as a baseline codebook. A $(36, 24)$ -code was developed to give a larger degree of freedom in selecting codewords; hence the number of sequences for the RCP was doubled with respect to the PBP. For the RCP with a $(36, 24)$ -code, the performance-based parameters are: $d_{\min}^H = 4$, $w = 4/24$, and $\text{TTI}_{\min} = 1$. Assuming a uniform distribution over characters, i.e. $P(C_m) = 1/M$, several $(36, 24)$ -codes were generated using different values for the performance-based parameters, with $\text{TTI}_{\min} \geq 3$, $d_{\min}^H \geq 4$ and $\omega \geq 4/24$ imposed.

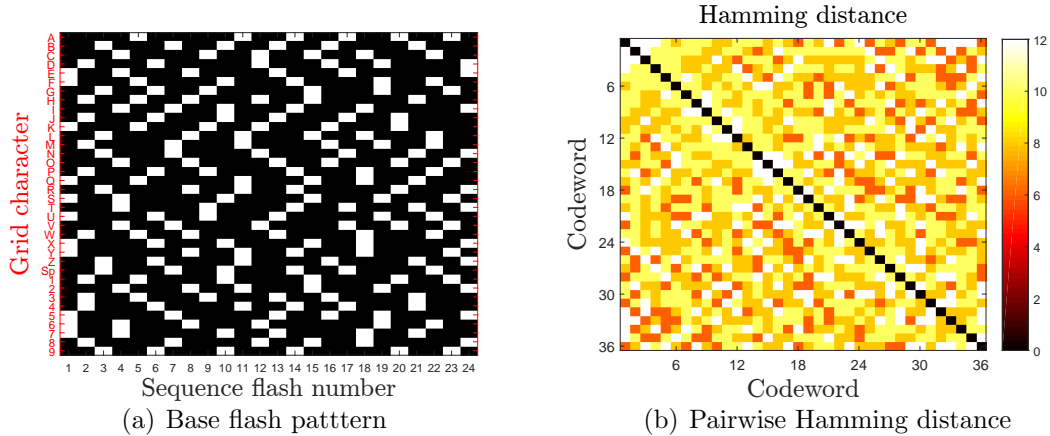


FIGURE 4.3: Performance-based paradigm for a $(36, 24)$ -code. (a) Codebook developed using Algorithm 5 with parameters $(M, l, \omega, d_{\min}^H, \text{TTI}_{\min}) = (36, 24, 0.25, 4, 3)$, assuming a uniform distribution over character choices. (b) Pairwise Hamming distance between codewords. Element (i, j) in the grid denotes the Hamming distance, $d^H(\mathbf{c}_i, \mathbf{c}_j)$, between codewords, \mathbf{c}_i and \mathbf{c}_j .

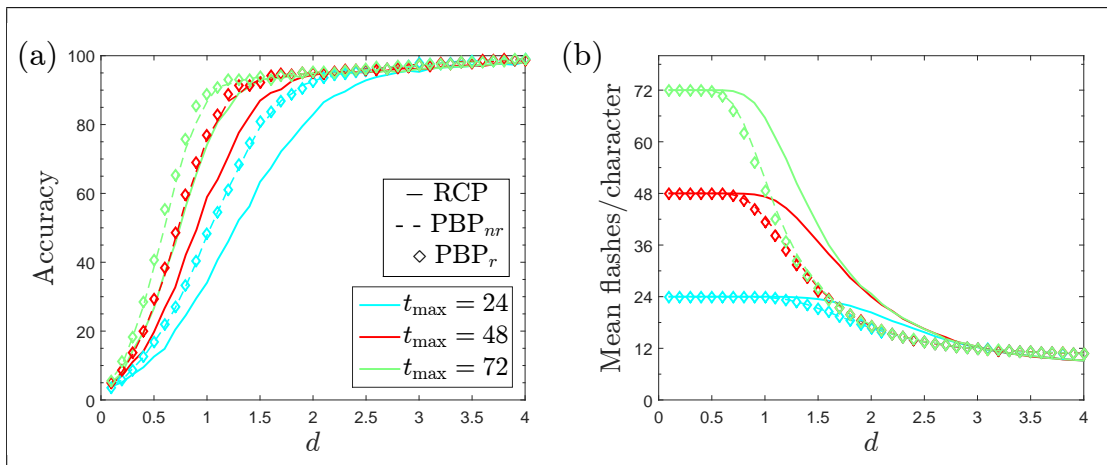


FIGURE 4.4: Performance vs. detectability index, d , for the row-column paradigm (RCP) and performance-based paradigm (PBP, Figure 4.3) using synthetic data: (a) accuracy and (b) expected stopping time. P300 spelling runs with the Bayesian dynamic stopping algorithm with $P_{th} = 0.9$ and data collection limit, t_{\max} , were simulated using normal distributions (for non-target and target classifier score likelihoods). For the PBP, the subscripts, nr and r , denote non-randomisation and randomisation, respectively, of character-to-codeword assignments after each sequence

Figure 4.3 shows the final selected codebook using Algorithm 5 with parameters $(M, l, \omega, d_{\min}^H, \text{TTI}_{\min}) = (36, 24, 0.25, 4, 3)$. The codebook for the 6×6 RCP (Figure

2.8(b)) is doubled to achieve the same codeword length as the PBP with a (36, 24)-code for comparison. The minimum target-to-target interval for the PBP ($\text{TTI}_{\min} = 3$) is higher than that of the RCP ($\text{TTI}_{\min} = 1$), and this is hypothesised to reduce TTI-related effects in the PBP. The target presentation frequency of the PBP ($\omega = 0.25$) is higher than the RCP ($\omega = 0.17$). The PBP has better Hamming distance properties, illustrated in Figure 4.3(b), with a $d_{\min}^H = 6$, compared to the RCP, Figure 2.8(c), with a $d_{\min}^H = 4$ if the base codebook of the RCP is doubled (2.6). It is hypothesised that the higher target presentation frequency and minimum Hamming distance of the PBP can result in improved accuracy by increasing the likelihood that the target character has the higher cumulative character response, as well as minimising decoding errors. The PBP also has a shorter average TTI ($\text{TTI}_{\text{avg}} \approx 4$) compared to the RCP ($\text{TTI}_{\text{avg}} \approx 6$), and this is hypothesised to reduce the EST for the PBP.

To demonstrate the potential performance improvements with the PBP, simulations of P300 spelling runs with normal distributions for the non-target and target classifier scores (Algorithm 4), were performed using the PBP and RCP and performance was compared. Two configurations of the PBP were tested: a non-randomised version, PBP_{nr} , where the character-to-flash group assignments were fixed, and a randomised version, PBP_r , where the character-to-flash group assignments were randomised after each sequence. Figure 4.4 shows the results from the simulations comparing the three stimulus paradigms. Both versions of the PBP outperform the RCP in terms of improved accuracy and reduced EST. No additional benefit was observed in the simulations for character-to-flash group randomisation as the performance for PBP_{nr} was the same as PBP_r .

4.2 Online Codebook Comparison Study

Since the new stimulus paradigm is developed with improved performance-based parameters, it is hypothesised that participant performance will be improved from the RCP. To verify the efficacy of the proposed codebook development method, the RCP and PBP are tested online.

4.2.1 Methods

Participants

The Duke University Institutional Board Review approved this study. Twenty healthy participants were recruited from the student and work population at Duke University, and they gave informed consent prior to participating in this study. Participants were numbered in the order they were recruited. All of the recruited participants were used for data analysis.

P300 Speller Task

Participants performed word copy-spelling tasks with the P300 speller in one session where the two stimulus paradigms were implemented using a 6×6 grid: the RCP and the PBP, shown in Figure 4.3. The word tokens were randomly selected from the English Lexicon Project [109]. Each participant session lasted about 1.5 -2 hours, including breaks. The calibration run involved data collection from five 6-character tokens (words or numbers), with no feedback presented to the user. Labeled data obtained from the calibration run were used to train a user-specific SWLDA classifier [33].

In the testing run, using the trained classifier, participants performed copy-spelling tasks of eight 6-character words with feedback and no error correction, except for participants 1 and 3 with five 6-character words. The Bayesian DS algorithm was used with probability threshold, $P_{th} = 0.9$. A data collection limit of $t_{\max} = 72$

flashes was imposed, equivalent to 6 sequences for the RCP and 3 sequences for the PBP. Each stimulus paradigm condition was calibrated and tested separately, with the order counter-balanced across participants.

The performance measures for comparison include: accuracy, the task completion time (in number of stimulus flashes and seconds) and bit rate. Statistical significance between the performance measures were tested using the Wilcoxon signed-ranked test, ($p < 0.05$).

4.2.2 Results

The performance measures determined from data obtained from the testing run are shown in Figure 4.5. Figure 4.5(a) shows the expected stopping time (EST), expressed in mean number of stimulus flashes prior to character selection. Participants used a significantly lesser amount of data flashes with the PBP (39.59 ± 14.21 flashes/character), compared to the RCP (51.74 ± 16.50 flashes/character), $p < 10^{-4}$. Figure 4.5(b) shows the amount of time per character selection, after taking into account the flash duration (62.5 ms), inter-stimulus interval (62.5 ms), and time pauses between character selections (3.5 seconds). Consequently, the amount of time per character selection significantly reduced with the PBP (8.45 ± 1.78 seconds) compared to the RCP (9.97 ± 2.06 seconds) d values, $p < 10^{-4}$.

Figure 4.5(c) shows participant accuracy. Despite a significant reduction in the amount of data collection with the PBP, for most participants, accuracy was either similar across paradigms or increased with the PBP. A significant improvement in accuracy was observed with the PBP (74.96 ± 18.15 %) compared to RCP (67.08 ± 22.51 %), $p < 10^{-2}$. Figure 4.5(d) shows participant bit rate, in bits per minute (bits/min). Due to maintaining similar or improving accuracy levels while reducing the amount of time per character selection, a significant improvement was observed in bit rate with the PBP (24.93 ± 12.50 bits/min) compared to the RCP ($18.94 \pm$

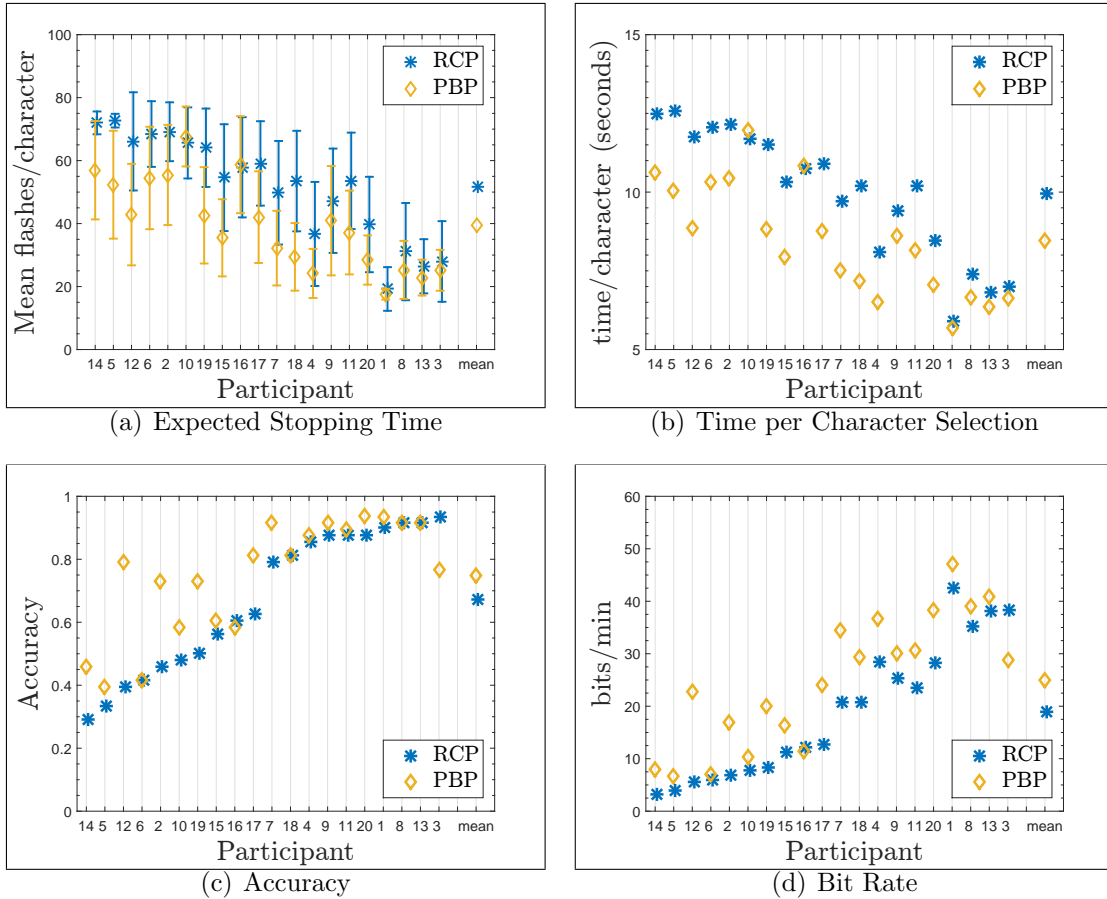


FIGURE 4.5: Online results comparing the row-column paradigm (RCP) and the performance-based paradigm (PBP, Figure 4.3), using Bayesian dynamic stopping with probability threshold, $P_{th} = 0.9$, and $t_{max} = 72$ stimulus flashes. Participant results are ordered by increasing RCP accuracy.

12.90 bits/min), $p < 10^{-3}$.

When developing the PBP, it was assumed that similar detectability index values would be observed across the stimulus paradigms to obtain the necessary improvements in performance from RCP to PBP. For each participant, d for the test data of both stimulus paradigms were calculated, and are shown Figure 4.6. Participant d values were more or less similar, with no significant differences between the PBP (1.37 ± 0.62) and the RCP (1.31 ± 0.55), $p = 0.15$. Based on the offline simulations, it was hypothesised that the PBP would result in a noticeable improvements in ac-

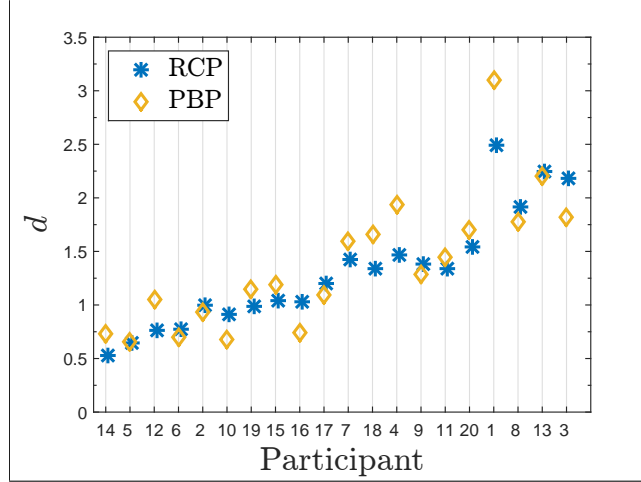


FIGURE 4.6: Comparison of participant detectability indices for the row-column paradigm (RCP) and performance-based paradigm (PBP). Participant results are ordered by increasing RCP accuracy.

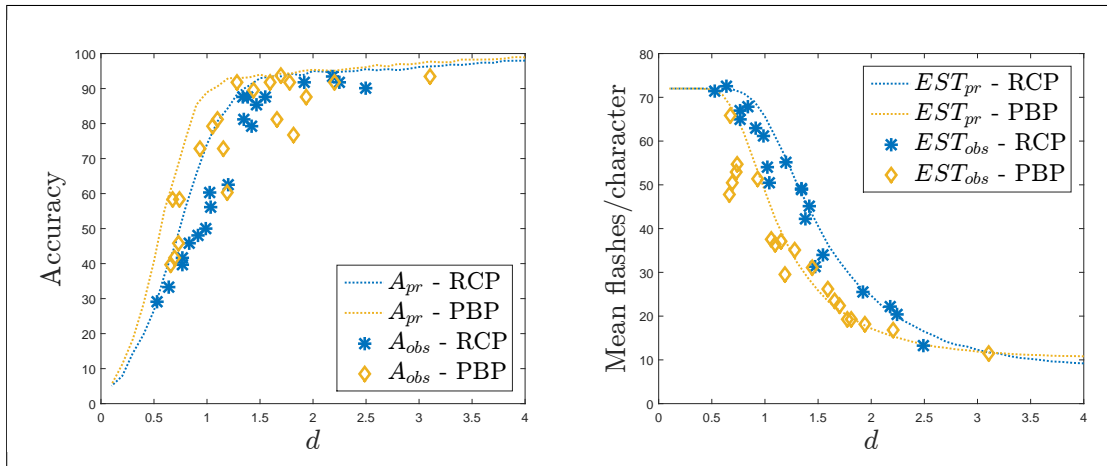


FIGURE 4.7: Performance vs. detectability index, d , for the codebook comparison study: row-column paradigm (RCP) and performance-based paradigm (PBP). The observed online performances are denoted, A_{obs} for accuracy, and EST_{obs} for the expected stopping time. The predicted performances, A_{pr} and EST_{pr} , were obtained from P300 speller simulations using normal distributions.

accuracy at low-range d values, and reduced EST at mid-range d values. Figure 4.7 compares the online participant performance to that predicted by the d values of the test data. The trends of the current results follow those predicted by the simulations, with improvements for accuracy and EST observed in the predicted range of d values.

4.3 Discussion

Previous stimulus paradigms that were designed with an information-theoretic approach did not result in performance improvements either due to an assumption of a memoryless P300 channel or a channel with short-term memory. In this work, it was hypothesised that the negative effects of successive target character presentation implied a P300 channel with a longer term memory for target stimulus events. While this long-term memory can be accounted for in a decoding algorithm, it is unlikely to result in significant performance improvements since the information content in repetitive character presentation is not useful.

Allowing for a “recovery” period between target character presentations is a more efficient means to improve performance by imposing a minimum interval between target stimulus events to mitigate TTI-related effects. Some stimulus paradigms adopt this approach by specifying a large minimum interval between target stimulus events. However, this decreases the BCI’s spelling rate, especially within the context of a dynamic stopping algorithm as it takes a longer time to converge to a decision threshold.

This chapter proposes a new method to develop a stimulus paradigm that focuses on tuning stimulus paradigm parameters to positively affect performance. Taking into account P300 channel dynamics, a performance-based paradigm is developed that minimises the expected stopping time of the Bayesian dynamic stopping algorithm. The codebook for the PBP is constructed via an iterative search of codewords that minimise the correlation of the character cumulative responses to reduce decoding errors. Based on empirical analysis, a shorter minimum target-to-target interval was chosen to achieve a compromise between improving spelling rate and minimising TTI-related effects. An added benefit of minimising TTI-related effects is the ability for simulation results to better reflect online trends. With online BCI use,

significant performance improvements, in both accuracy and spelling speed, were obtained with the performance-based paradigm compared to the conventional row-column paradigm. It is shown that BCI performance can indeed be improved across a wide range of performance levels by using an information-theoretic approach with the proper P300 channel dynamics.

Increasing Language Model Complexity in the Target Character Estimation Algorithm

The predictability of language is exploited in many text-entry applications to enhance user experience by increasing system throughput, e.g. predictive text, auto-correct, machine translation etc. Similarly, language information has previously been incorporated into BCI communication devices with positive results [110]. The complexity of the language information can be increased to facilitate BCI use towards more practical communication with phrases or sentences. However, integration of language information into BCI spellers requires additional consideration of the limitations experienced by people with severe disabilities when navigating the user interface. Some methods of integrating language information involve a changing user interface with predictive word suggestions where a user can select one of the options if desired, e.g. [85, 86]. However, this approach can be susceptible to increased task difficulty due to the additional cognitive load required when adapting to a changing visual interface. Nonetheless, there is still potential to improve speller performance by designing a less cumbersome interface to minimise any increase in cognitive load, e.g. [87].

Alternatively, language information can be integrated within the target character estimation algorithm to enhance the data collection process, with minimal or no change to the user interface [60, 111]. It was hypothesised earlier that the inclusion of a letter bigram model in the probability intialisation step of the Bayesian DS algorithm might be too simplistic to result in significant improvements in accuracy in the previous ALS study in [88] since the language model exploits only a limited amount of the user’s spelling history. A higher order language model has the potential to better utilise language information in the target character estimation process but its utility depends on correct selection of previous target characters. This work proposes a further extension of the language model in the Bayesian DS algorithm to include more of the user’s spelling history while minimising the impact of erroneous character selections on the language model.

5.1 Higher Order Language Models

As more letters of a word are spelled, the set of subsequent likely letters narrows. This narrowing down of likely letters is better captured by a higher order language model. For example, if a user spells **EP**, it is highly likely that the subsequent letter is a vowel or the consonants **S** or **H**, depending on the context. However, the utility of higher order models is dependent on the number of preceding characters that are correctly spelled to generate the correct likely prefix in order to determine the initialisation probabilities of the subsequent characters. To reduce the impact of misspelled characters on the language model probabilities, Mainsah *et al.*, (2014a) [59] introduced an error factor in the character initialisation process via a weighted combination of a bigram model and a uniform distribution (2.12). In Orhan *et al.*, (2012) [60] where a 6-gram model was used, all erroneous character selections had to be revised.

Rather than rely on an error factor parameter to compensate for misspelled char-

acters without considering multiple options or force the user to correct all misspelled characters, a new approach to account for possible misspellings is proposed in order to improve the utility of a higher order n -gram model. This new approach relies on the character likelihoods obtained from the target character estimation process of previous character selections to generate a set of likely word prefixes. These word prefixes are used to develop a language model, where each prefix contribution to the model is weighted by their respective cumulative character likelihoods. This prefix determination process is combined with a dictionary look-up to ensure valid word prefixes. In this study, the performance of the proposed method is compared to the performance of the previous Bayesian DS algorithm with a bigram model [59], and another language-based dynamic stopping algorithm developed for the P300 speller [81].

5.1.1 Methods

EEG Dataset

The dataset was obtained at Duke University from a healthy participant study approved by the Duke University Institutional Review Board. Twenty participants were recruited from the student and work population at Duke University. All participants gave informed consent prior to participating in this study. Participants were numbered in the order that they were recruited.

Each participant participated in a P300 spelling session consisting of two phases: a calibration run to collect data to train the P300 classifier and a test run to perform online copy-spelling tasks. Off-line analysis on the test dataset was performed using MATLAB software (The MathWorks, Inc.) to simulate P300 spelling runs with different language-based dynamic stopping algorithms. The language model was developed from a corpus compiled by Norvig [112].

Data Collection Algorithms

(i) Bayesian DS with a Weighted n -gram Model

The order of the language model in the probability initialisation step of the Bayesian DS with a language model (DSL_M) [59] can be modified to include more of the user’s spelling history: $P(C_m | \mathcal{A}_{T-(n-1)}^{T-1})$ in (2.12). If considering all of a user’s previous character selections, then $n = T$, i.e. $\mathcal{A}_{T-(n-1)}^{T-1} = a_1, \dots, a_{T-1}$. \mathcal{A}_1^{T-1} denotes a prefix which is used to determine likely words to derive the language model probabilities. However, when using the n -gram model with all of the user’s spelling history, there is the possibility of generating an invalid vocabulary prefix if an erroneous character is selected, (e.g. **VIS2** for the word **VISUAL**), unless the erroneous character is revised.

One way to possibly obtain the correct prefix is to use the Viterbi algorithm [113] to determine the most likely sequence of characters based on letter transitional probabilities and the character likelihoods from the target character estimation process, as illustrated in Figure 2.12. In this work, these character probabilities are denoted as a $\mathbf{Q}^{N \times T}$ matrix, where N is the number of grid characters and T is the number of previously selected characters. However, the Viterbi algorithm is unlikely to correct erroneous character selections when letter transitions are highly likely. For example, if a user spells **ANC** for the word **INCOME**, the Viterbi algorithm will still select **ANC** since the letter transitions are highly likely, and this incorrectly biases the language model. The Viterbi algorithm can be further expanded to generate alternative prefixes by using a k -best algorithm that generates the k most likely prefixes [114]. The k -best Viterbi algorithm was initially considered in this work. However, preliminary offline analysis revealed that there was no guarantee that the prefixes generated by the method will be valid dictionary prefixes as a limited neighbourhood of characters is considered when using the language model. Thus, Viterbi was

discarded as the prefix selection method and a new method was developed.

As an alternative, the \mathbf{Q} matrix can be used directly in conjunction with a dictionary query to generate a set of highly likely, yet valid prefixes. These prefixes can be weighted by their combined character likelihoods when determining the character initialisation probabilities. Let \mathcal{D}_T^k denote a set of k valid prefixes of length $T - 1$ with the top k combined character likelihoods. \mathcal{D}_T^k is used to derive the language model probabilities for the probability initialisation step to select the current target character, C_T^* . These prefixes are retained from one character to the next to generate the next k most likely prefixes.

The conditional probabilities in the initialisation step to select the current target character now depend on the previous character selections, \mathcal{A}_1^{T-1} , and the set of k highly likely prefixes, \mathcal{D}_T^k determined from a \mathbf{Q}_{T-1} matrix:

$$w(D^j) = \frac{\prod_1^{T-1} q_t^{l(D_t^j)}}{\sum_{D^j \in \mathcal{D}_T^k} \left(\prod_1^{T-1} q_t^{l(D_t^j)} \right)} \quad (5.1)$$

$$P(C_i = C_T^* | \mathcal{A}_1^{T-1}, \mathbf{Q}_{T-1}) = \sum_{D^j \in \mathcal{D}_T^k} w(D^j) P(C_i = C_T^* | D^j) \quad (5.2)$$

where $w(D^j)$ is the weight of the prefix D^j ; $l(D_t^j)$ is the label for the t^{th} character in prefix D^j ; $P(C_{i,T} | D^j)$ is the conditional probability that the T^{th} target character is C_i , given prefix D^j . The weighted language model generated by the above method will be referred to as the n -gram model with dictionary-assisted prefix search, n -gram DAPS.

(ii) *Forward Algorithm*

In this study, the proposed algorithm was also compared with another language-based data collection algorithm proposed by Speier *et al.*, (2014) [81]. Their method uses the forward algorithm as the target character estimation algorithm (Appendix

E) to control data collection. Similar to the Bayesian DS algorithm, data collection is stopped when the forward probability of a character attains a preset threshold probability and that character is selected as the user’s intended target.

There are several differences between the Bayesian n -gram DAPS algorithm and the forward algorithm used in [81]. The forward algorithm uses a trigram model, i.e. the initialisation probabilities depend on the previous two character selections. However, the n -gram DAPS model uses all of the user’s spelling history. Also, the two algorithms utilise the \mathbf{Q} matrix differently. The initialisation probabilities in the forward algorithm depend on all possible sequences of letters. However, for the n -gram DAPS model, only the k most likely and valid prefixes contribute to the initialisation probabilities; since the \mathbf{Q} matrix is generally sparse, the contribution from each additional prefix decreases substantially.

5.1.2 Results

The results include algorithm performance evaluations based on an offline analysis of EEG data and an online participant study. In the offline analysis, P300 speller simulations were performed on participant EEG data using three data collection algorithms: the forward algorithm with a trigram language model, and the Bayesian DS algorithms with a bigram and n -gram DAPS model. In the online study, only the Bayesian DS algorithms were implemented and compared.

Offline Analysis

Participant EEG data were used to simulate dynamic data collection with the forward algorithm with the trigram model and the Bayesian DSLM algorithm with bigram and n -gram DAPS models. Participants were evaluated according to the following performance measures: accuracy, expected stopping time and theoretical bit rate. Statistical significance was tested using a repeated measures ANOVA, with

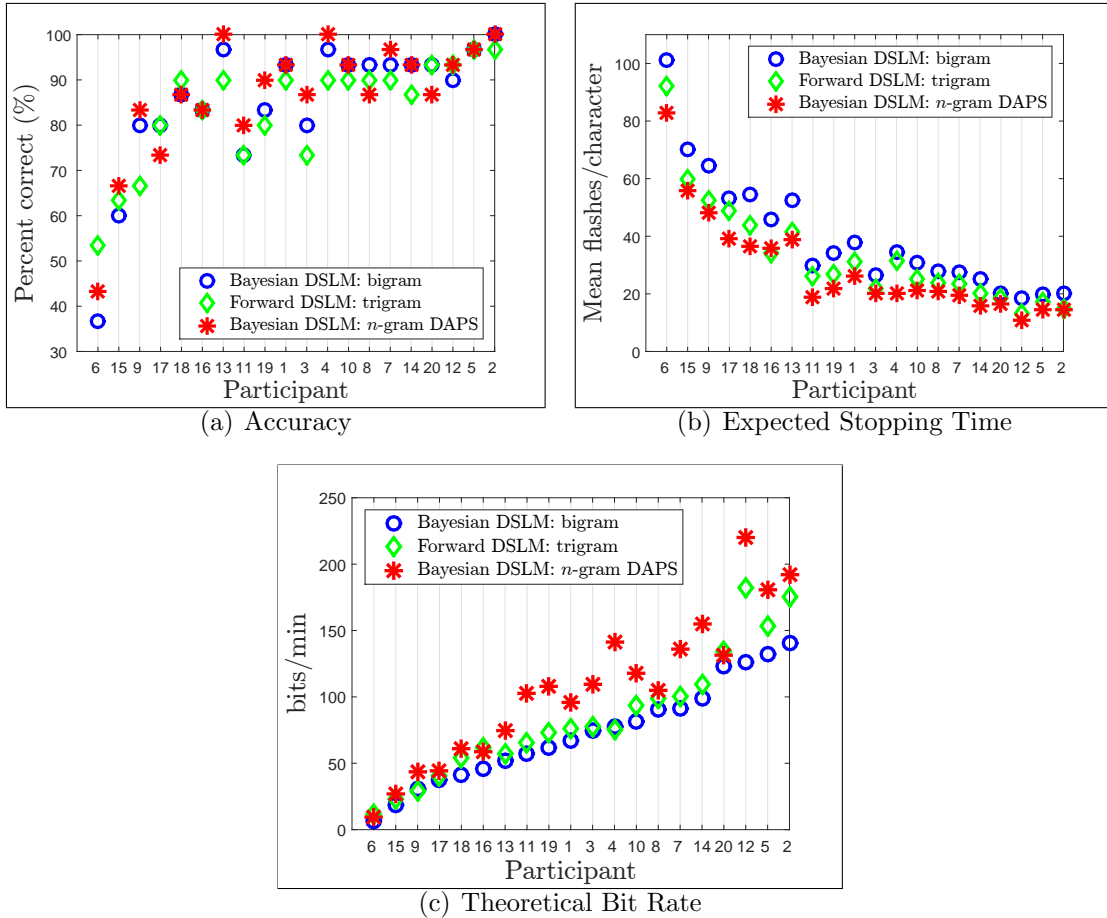


FIGURE 5.1: Offline performance of different dynamic stopping algorithms with language models (DSLML): (a) accuracy, (b) expected stopping time, and (c) theoretical bit rate. P300 speller simulations were performed using participant EEG data using three algorithms: the Bayesian dynamic stopping (DS) algorithm with bigram and n -gram dictionary assisted prefix search (DAPS) models, and the forward algorithm with trigram model. Participants are ordered by increasing theoretical bit rate with the Bayesian DS algorithm with a bigram model.

additional pairwise comparisons (Bonferonni adjustment) if the test revealed a significant difference in the means of at least two algorithms ($p < 0.05$).

Figure 5.1(a) shows each participant’s selection accuracy for the three data collection algorithms enhanced with a language model. On average, accuracy improved from the bigram to the trigram to the n -gram DAPS algorithm, but the differences between the algorithms were not statistically significant, $p = 0.067$. The \mathbf{Q} matrix

is used in the n -gram DAPS model to weight only valid prefixes. Most participants observed an improvement in accuracy from the bigram to the n -gram DAPS model. The \mathbf{Q} matrix tends to be sparse, meaning that only a few characters are considered likely to be the target character by the algorithm. These characters typically have at some point been in the same flash group as the target as this is often the source of most of the character selection errors. It is hypothesised that the more informative priors introduced by a higher order language model give the target character an added advantage to prevent an erroneous character selection, thereby sometimes improving accuracy.

Figure 5.1(b) shows the expected stopping time for each participant. In general, character selection time reduced with increased order of the language model used, with a statistically significant decrease in stopping time observed from bi-gram to trigram to n -gram DAPS model, $p < 10^{-5}$. In Mainsah *et al.* (2014), off-line analysis of participant EEG data revealed that the rate of convergence to the threshold probability with the Bayesian DS algorithm increased with the inclusion of the bi-gram language model [59]. The more informative priors introduced by higher order language models further increase the rate of algorithm convergence. Hence, the observed consistent reduction in character selection times across all participants with increased language model order.

Figure 5.1(c) shows participant theoretical bit rate, in bits per minute, which excludes the time pauses between character selection. Due to similar accuracy levels and a significant reduction in character selection time, most participants observed statistically significant improvements in their theoretical bit rates from the Bayesian bigram (72.73 ± 38.30 bits/min) to forward trigram (84.70 ± 47.63 bits/min) to the Bayesian n -gram (105.65 ± 56.13 bits/min) algorithms, $p < 10^{-5}$.

Online Performance

For the online study, participants performed copy-spelling tasks, with no error correction, using the Bayesian DS with the bigram and n -gram DAPS language models [115]. Algorithm order was counter-balanced across participants to avoid biasing the results. Figure 5.2(a) shows each participant’s selection accuracy. There was no significant difference in participant accuracy between the bigram ($86.64 \pm 17.06\%$) and the n -gram model ($90.650 \pm 11.96\%$), $p = 0.09$. However, noticeable improvements were observed for the low performing participants (participants 6 and 15).

The n -gram DAPS model is still susceptible to errors if enough prefixes are not retained to potentially contain the target prefix or if other prefixes have stronger priors than the target prefix. For example, participant 10 experienced a drop in accuracy with the n -gram DAPS model entirely due to the BCI selecting characters **AFT-D** while trying to spell **AVOID**, as **AF** is a valid dictionary prefix likely to be followed by **T**. It should be noted that in this study, unigram word probabilities, i.e. derived from word frequency, were used in developing the language model as the spelling tasks involved single words. Providing additional context via bigram or trigram word probabilities in sentences might better refine the character initialisation process by minimising the effect of highly likely but non-target words via their priors.

Figure 5.2(b) shows the average amount of time per character selection for each participant. A statistically significant decrease in character selection time was observed with the n -gram DAPS model (4.64 ± 1.11 minutes) compared to the bigram model (5.04 ± 1.45 minutes), $p = 0.002$. Figure 5.2(c) shows participant bit rates, which includes the time pauses between character selections. Due to similar accuracy levels and a significant reduction in character selection times, most participants observed significant improvements in their performance from the bigram (32.65 ± 13.25 bits/min) to the n -gram model (36.50 ± 12.40 bits/min), $p = 0.007$.

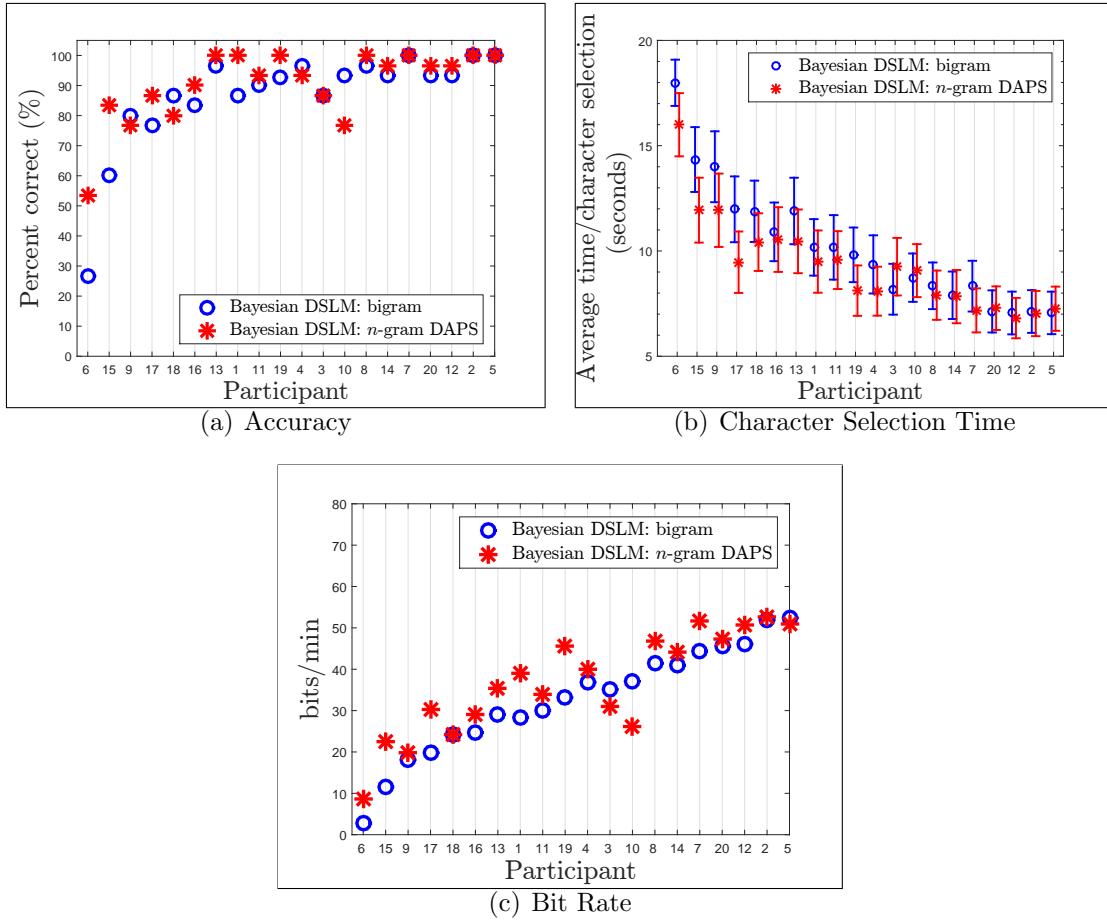


FIGURE 5.2: Online performance for the Bayesian dynamic stopping algorithms with different order language models (DSLML): bigram and n -gram dictionary assisted prefix search (DAPS). Participants are ordered by increasing bit rate with the Bayesian DS algorithm with a bigram model.

The performance improvement from the bigram to the n -gram model in this online study are consistent with trends from the previous off-line analysis.

5.2 Discussion

The relatively slow communication rates of ERP-based BCI speller systems can be improved by exploiting the predictability of language. There is the potential to improve BCI performance by increasing the order of the statistical language model that is incorporated into the target character estimation algorithm. In this study, addi-

tional consideration was taken to minimise erroneous character revisions with higher language model order by using the n -gram model in combination with a dictionary. In Mainsah *et al.*, (2014a), most of the improvements with the bigram model were observed in the mid-range performing participants [59]. In this study, noticeable improvements in accuracy were observed in the lower performing participants with the more complex language model. Thus, it is possible to improve performance and expand the pool of potential BCI users by extending the language model in the target character estimation algorithm.

Error Correction in P300 Spellers

To be an effective communication tool, it has been recommended that ERP-based BCI spellers perform with at least 70% accuracy [12] to account for revisions of erroneous selections. Alternatively, system usability can be improved if errors can be automatically detected and deleted without further user action. Error related potentials (ErrP), which are EEG-based responses elicited as a result of errors, have been proposed for error detection. However, ErrP-based correction comes with appreciable costs in terms of additional data required and time to collect the data.

There are various tools available in many language-based applications which can infer a user's intended character/word based on the spelling history and *a priori* language information, e.g. spell-check, auto-correct, predictive spelling. However, individuals with severe disabilities have limited ability to easily navigate choices within a changing user interface, and this has to be taken into account when integrating such error detection/correction strategies in BCI systems. This chapter proposes a language-based spelling correction method for the P300 speller, that is as effective as a perfect ErrP classification system without the associated costs, by exploiting information that is already available from the P300 character selection

process.

6.1 Error Information in P300 Classifier Outputs

Erroneous character selections in the P300 speller typically have some predictability as classification confusions are most likely to consist of characters that are in the same flash group as the target character. For example, error analysis of P300 speller selections using the RCP revealed that most erroneous character selections were usually in the same row or column as the target character [61, 50], and for spellers using the CBP, most erroneous character selections were usually in the same virtual matrix [50]. The structure of erroneous character selections is reflected in the cumulative P300 classifier outputs. Prior to character selection, the cumulative P300 classifier outputs contain information about the likelihood of characters being the target at each position in a word [116].

Figure 6.1 shows two example character probability distributions post-data collection for the word **DRIVING**, obtained from P300 speller simulations using EEG data from two different participants, to yield the words **-J I, S-G** and **-R-VING**, respectively. A 9×8 CBP P300 speller grid was used. The command options (e.g. “Del”, “Tab”, “Sleep” etc.) were disabled, and if selected, were represented with a hyphen. The character probability distribution post-data collection is denoted as the **Q** matrix. The **Q** matrix is usually sparse as there are only a few characters in each flash group. The level of sparsity increases with increased user performance level, e.g. the distribution of the user with more correct character selections, distribution 2, is sparser.

Ideally, the target character should have the highest probability value post-data collection. However, it can be noted that for erroneous character selections in both probability distributions in Figure 6.1, the target characters are usually among those with the next highest probabilities. The character probabilities can thus be used to

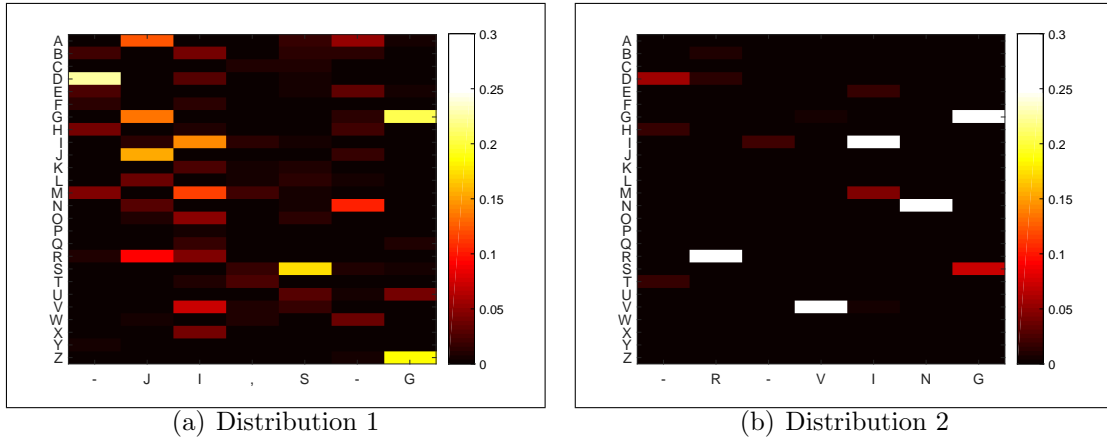


FIGURE 6.1: Distribution of Bayesian character probabilities post-data collection for the word **DRIVING**, obtained from simulations of P300 spelling runs with EEG data from two user copy-spelling sessions. The x -axis labels show the characters selected by the P300 speller simulation, yielding the words **-JI,S-G** (left) and **-R-VING** (right), with the corresponding probabilities of alphabet characters which are shown in the y -axis. Probability values are clipped for visualisation and non-alphabetic characters are not displayed.

assess the likelihood of each character at each word position, e.g. character **D** in both distributions has the highest alphabet probability given the erroneous selection at position 1. This information can be useful in language-based applications such as further enhancing the character initialisation step in the Bayesian dynamic data collection or doing dictionary-based spelling correction.

The cumulative P300 classifier outputs can be used to narrow down multiple word choices obtained from a dictionary via string matching, e.g. [94]. For example, for the spelled word **-R-VING**, some word alternatives include CRAVING, DRIVING, PROVING, etc. However, the number of word choices increases as the number of errors increases, as in the spelled word **-JI,S-G**. The character probabilities, such as the Q matrices shown in Figure 6.1, can be used to weight candidate word choices when performing dictionary-based spelling correction.

6.2 Preliminary Analysis of Error-Correction Mechanisms

Conventionally, an erroneous selection by a BCI system requires at least two corrective actions: an action to undo the previous erroneous selection and another action to re-select the intended choice. In active user correction, these actions are initiated by the user. With an ErrP-based correction mechanism, selecting the deletion command can be bypassed by having the BCI automatically delete the erroneous selection upon ErrP detection. However, there is the possibility of making errors in the correction process, which can lead to additional increases in spelling time. For example, the BCI can incorrectly select another character instead of a *backspace* command, or the ErrP classifier can erroneously flag a correctly selected character for deletion. Predicting the resulting increase in spelling time is important to assess the potential benefit that might result from any error-correction mechanism. The active user and ErrP-based correction mechanisms are here analysed to evaluate their potential benefit to a BCI user.

6.2.1 Methods

Active User Correction

In active user correction, after an erroneous BCI selection, a user initiates both the selection of a deletion command and the re-selection of the intended character. Assuming a P300 speller system with probability, p_e , of making an erroneous selection (Accuracy, $A = 1 - p_e$) and accounting for deletion and re-selection actions, the

effective number of selections needed to correctly make N selections is [50]:

$$E[N] = N + 2p_e N + 2p_e(2p_e N) + \dots \quad (6.1a)$$

$$= \sum_{i=0}^{\infty} N(2p_e)^i \quad (6.1b)$$

$$= \frac{N}{1 - 2p_e}, \text{ with convergence} \quad (6.1c)$$

which is an infinite geometric series that converges to (6.1c) if $p_e < 0.5$. The above estimate provided in [50] assumes that a *backspace* character or deletion command is not selected accidentally, although this is a possibility in real world conditions. However, if the number of character choices is large, the probability of accidentally selecting a *backspace* command is small and (6.1c) can provide a close estimate.

ErrP-based Correction

Unlike with the standard P300 speller where a character selection is made after a certain amount of data collection, ErrP-based correction is based on a single trial detection of an ErrP following the presentation of the BCI feedback. An ErrP is detected if the classifier score is above or equal to a threshold value, denoted as τ . If an ErrP is detected, the current BCI selection is vetoed. The performance of an ErrP-based correction mechanism is dependent on selecting an appropriate operating point on the ErrP classifier receiver operating curve (ROC) which minimises the probability of an erroneous decision. The operating point consists of a pair of values, a probability of detection (P_D) and a probability of false alarm (P_{FA}) on the ROC curve. However, this approach does not consider the outcome of the P300 speller system. Every auto-deletion with ErrP detection requires re-selection of a character by the P300 speller system. Hence, the benefit of selecting an ErrP classifier operating point has to be evaluated within the context of the accuracy of P300 speller

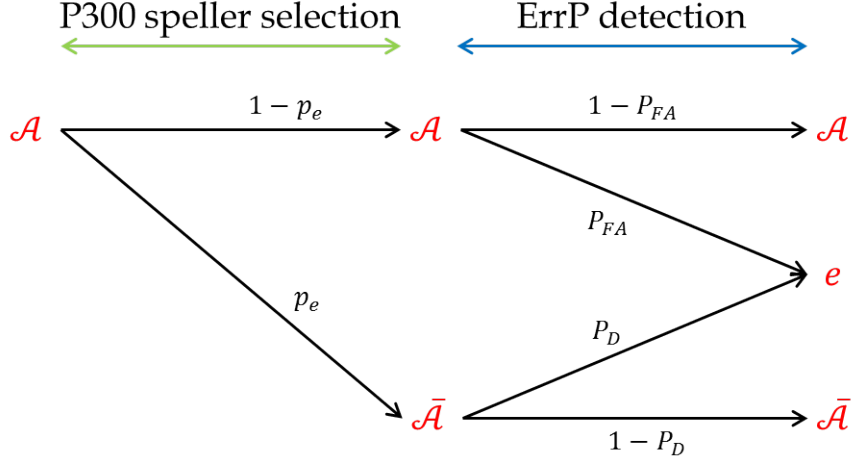


FIGURE 6.2: ErrP corrective mechanism following P300 speller selection. The P300 speller selects a character the user intends to spell, \mathcal{A} , with probability of error, p_e . Following feedback presentation to the user, an ErrP classifier with probability of detection, P_D , and probability of false alarm, P_{FA} , is used to flag characters if an ErrP is detected. The detection of an ErrP results in an automatic character deletion or an erasure, e .

selections.

This work proposes an approach to select an operating point for the ErrP classifier that minimises the expected number of selections with erroneous character revisions, taking into account the accuracy of the P300 speller system. Still considering a P300 speller with p_e , taking into account erroneous character revisions, and now using an ErrP classifier with operating point (P_D, P_{FA}) for automatic character deletions, the possible chain of events are illustrated in Figure 6.2. For the P300-ErrP cascade, (6.1) can be modified to:

$$\mathbb{E}[N] = N + N [2p_e(1 - P_D) + (1 - p_e)P_{FA} + p_eP_D] + \dots \quad (6.2a)$$

$$= \sum_{i=0}^{\infty} N [2p_e(1 - P_D) + (1 - p_e)P_{FA} + p_eP_D]^i \quad (6.2b)$$

$$= \frac{N}{1 - [p_e(2 - P_D) + (1 - p_e)P_{FA}]}, \text{ with convergence} \quad (6.2c)$$

where in (6.2a): $p_e(1 - P_D)$ corresponds to the proportion of P300 speller selections requiring two selective actions to correct erroneous character selections not detected by the ErrP classifier; $(1 - p_e)P_{FA}$ corresponds to the proportion requiring one selective action to re-select incorrectly flagged and deleted correct character selections by the ErrP classifier; and p_eP_D corresponds to the proportion requiring one selective action to re-select correctly flagged and deleted erroneous character selections. The series converges if $p_e(2 - P_D) + (1 - p_e)P_{FA} < 1$. The range for effective BCI communication can potentially be increased beyond requiring a selection accuracy of 70% by adjusting the operating point (P_D, P_{FA}) of the ErrP classifier accordingly.

6.2.2 Results

To examine the potential changes in spelling time, P300 spelling runs were simulated with both error-correction mechanisms. In the simulations, all erroneous character selections had to be revised until all of the intended characters were correctly selected. Also, a backspace character was allowed to be accidentally selected. The total number of selections needed to achieve 100% accuracy was determined for each set of simulations and compared to the respective theoretical estimates.

The first experiment involved simulating active user correction, with the simulation parameters obtained from participant EEG data from P300 spelling sessions, in order to confirm the predicted estimate in (6.1c) [50]. The results from the simulations are shown in Figure 6.3. From the results, the total number of selections to achieve 100% accuracy, $E[N]$, obtained via simulations closely match the derived theoretical estimates obtained from (6.1c). It can also be observed that $E[N]$ increases substantially as p_e approaches 0.5. This demonstrates why P300 speller use is not indicated for users with low accuracy levels, as the process for correcting erroneous BCI selections will take these users an excessive amount of time.

In a second experiment, P300 speller selection was cascaded with ErrP detection,

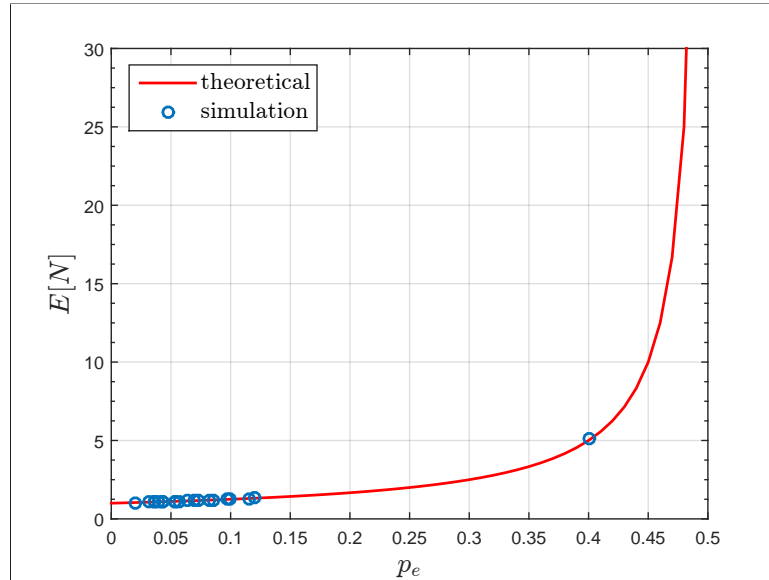


FIGURE 6.3: Effective number of P300 speller selections, $E[N]$, vs. probability of error, p_e , of P300 speller selection. P300 speller simulations of copy-spelling tasks were performed using participant EEG data. Selections and revisions were allowed to occur until all the intended characters were correctly selected.

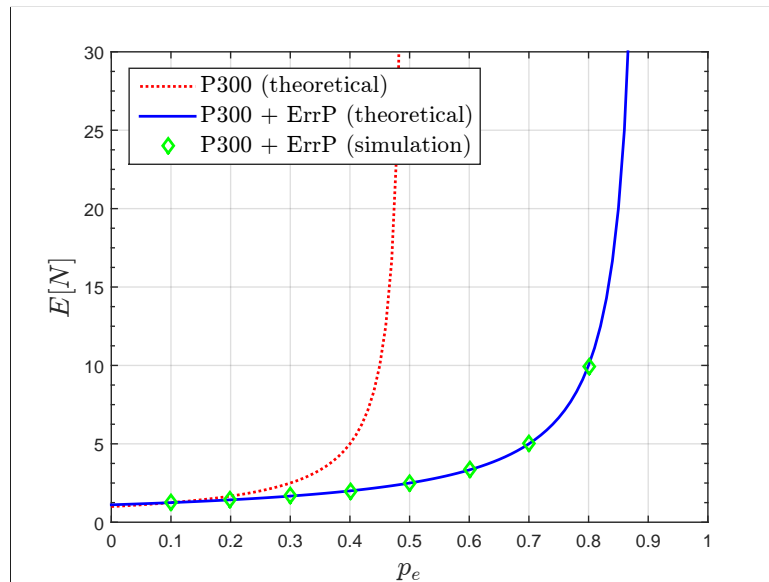


FIGURE 6.4: Effective number of P300 speller selections, $\mathbb{E}[N]$, vs. probability of error, p_e , of P300 speller selection with and without an ErrP cascade. The operating point for ErrP classifier was set to $P_D = 0.9$ and $P_F = 0.1$. P300 speller simulations of copy-spelling tasks were performed. Following feedback presentation, the ErrP classifier is used to flag an erroneous character if an ErrP is detected. The detection of an ErrP results in an automatic character deletion. P300 speller selections and revisions were allowed to occur until all the intended characters were correctly selected.

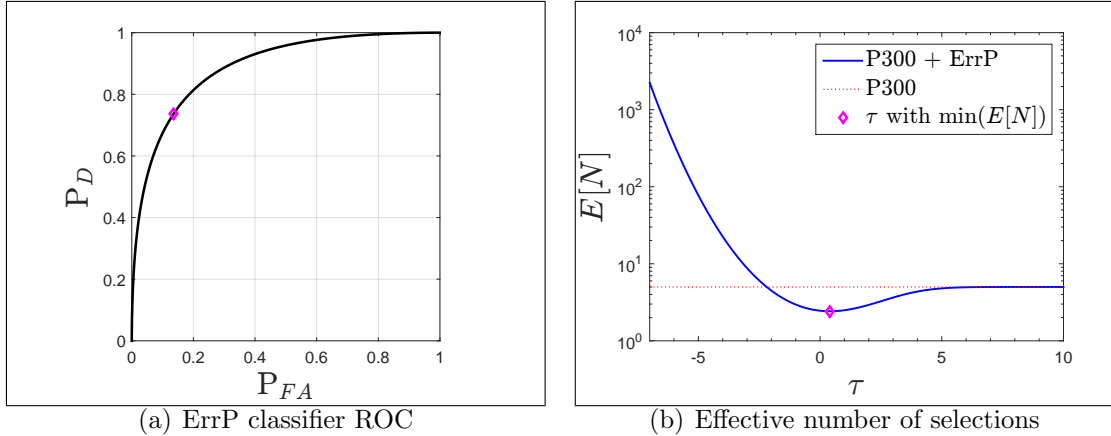


FIGURE 6.5: Selection of an optimal operating point for the ErrP classifier in a P300-ErrP cascade. (a) Illustrative example of a receiver operating characteristic (ROC) of an ErrP classifier. An ErrP is detected if the classifier score is above or equal to a threshold value, τ . P_D and P_{FA} are the probabilities of correction detection of an ErrP, and false alarm or incorrect detection of an ErrP. (b) Theoretical estimate for the effective number of selections, $E[N]$, with a correction mechanism: P300-only and P300-ErrP cascade. P300 speller selection has a probability of error, $p_e = 0.4$ and $E[N]$ is determined from (6.1c). For the P300-ErrP cascade, (p_e, P_D, P_{FA}) , $E[N]$ is determined from (6.2c), and the τ that minimises $E[N]$ can be obtained.

as illustrated in Figure 6.2. Figure 6.4 shows results from simulating the P300-ErrP cascade with one operating point of an ErrP classifier across a range of p_e values. The results from the simulations are consistent with the theoretical estimate derived in (6.2c). This verifies the proposed metric to evaluate performance if deciding to use ErrP-based correction. Also, the benefit of ErrP detection and automatic character deletion is evident at higher p_e values when comparing $E[N]$ from the P300-ErrP cascade (6.2c) to that of only a P300 speller selection process (6.1c).

From (6.2c), it can be observed that $\mathbb{E}[N]$ for the ErrP-based correction mechanism now depends on three parameters: p_e, P_D and P_{FA} . Assuming a P300 speller with fixed p_e , an operating point, (P_{FA}^*, P_D^*) , on the ROC of an ErrP classifier can be found that minimises $\mathbb{E}[N]$ in (6.2c), or for computational simplicity, maximises the denominator, $1 - [p_e(2 - P_D) + (1 - p_e)P_{FA}]$. To demonstrate the effect of selecting a suitable operating point on the ROC of the ErrP classifier, the performances of

P300-only correction and that of a P300-ErrP cascade are compared. A possible ROC of an ErrP classifier is shown in Figure 6.5(a). The theoretical estimates for $E[N]$ for both mechanisms are shown in Figure 6.5(b). It can be observed that either of the correction mechanisms outperforms the other at different regions of the ErrP classifier ROC. In the region where the P300-ErrP cascade outperforms, an operating point on the ErrP classifier that minimises $E[N]$ can be obtained.

In active user and ErrP-based correction, every erroneous BCI selection has to be corrected. However, since the P300 speller is a communication device, this might not be necessary. The correction process can be delayed to correct spelling errors in words by using prior language information. The previously mentioned error correction methods fail to exploit the predictability of language. Language-based spelling correction methods, like the noisy channel model (2.13-2.14) have a better potential to improve P300 speller accuracy as prior language information can be used to achieve error detection and/or correction.

6.3 Comparison of Spelling Correction Methods

ErrP-based correction has been proposed for P300 speller correction and the few online studies which have implemented such methods have yielded mixed results [27, 28, 91, 92]. Given the limitations of ErrP-based correction, alternative approaches like language-based methods for correction, have a better potential to enhance P300 speller performance [94]. Instead of character-based correction, this work proposes word-based correction with a noisy channel model. The noisy channel model can be based on the ErrP classifier outputs or the P300 classifier outputs. Thus, in addition to developing the noisy channel model independent of ErrP detection, a noisy channel model is derived based on ErrP classifier outputs to verify that any advantage of relying on P300 classifier outputs for error correction is due to the value of the information, not the use of the noisy channel model. In this study,

non-ErrP based correction methods were compared against ErrP-based methods to see if similar improvements to P300 speller performance could be obtained.

6.3.1 Methods

The non-ErrP correction methods consisted of a dictionary look-up with different word selection methods: word frequency, a method proposed by Ahi *et al.* (2011) [94] for word ranking, and a Bayesian method based on P300 speller character probabilities. The ErrP detection based methods include: an “oracle” method in which perfect ErrP detection was assumed, a method proposed by Perrin *et al.* (2012) [91] for using ErrP detection for error correction, and a method for which the ErrP detection classifier confidences are used in the noisy channel model. The noisy channel model with ErrP and P300 classifier confidences are new techniques developed for this study. Off-line signal analysis and spelling correction were performed using MATLAB software (The MathWorks, Inc.).

EEG Dataset

The dataset was obtained at East Tennessee State University for a study approved by the East Tennessee State University Institutional Review Board. Participants were numbered in the order they were recruited. Participants underwent two P300 speller sessions: a first session to collect data to train a P300 classifier and a second to collect data to train an ErrP classifier. During the first session, participants spelled four 5-letter words with five sequences/character. During the second session, the trained P300 classifier was not used online. However, participants spelled 15 phrases of 20 characters, each with fake feedback presented at an error rate of 20%. To speed up data collection for the ErrP classifier, only one sequence/character was used prior to presenting the fake feedback.

P300 Classification

EEG data from the first training session (no feedback, 5 sequence/character) were used to train a SWLDA classifier [33]. The data were also used to generate the likelihood probability density functions (pdf), $p(y|H_0)$ and $p(y|H_1)$, of target and non-target P300 classifier scores, respectively, via kernel density estimation (KDE).

ErrP Classification

Using the EEG data post-character feedback from the second training session (fake feedback at 20% error rate, 1 sequence/character), features were extracted to train a linear discriminant analysis (LDA) classifier with shrinkage [35], using leave-one-word-out cross-validation. The likelihood pdfs, $p(s|H_c)$ and $p(s|H_e)$, of correct and erroneous character scores, respectively, were generated via KDE and used for the ErrP-based spelling correction algorithms.

P300 Spelling and ErrP Detection Simulations

For each subject in the dataset, the P300 classifier trained from the first spelling session data was applied to the EEG data of the second session to simulate P300 spelling. In the example shown in Figure 6.6, a user intends to spell the word $C = (c_1, c_2, \dots, c_T)$. Using the P300 speller, the user spells the word, $W = (w_1, w_2, \dots, w_T)$, where character w_t , is the character that maximises the cumulative P300 classifier score function. The P300 classifier also outputs a $\mathbf{Q}^{N \times T}$ matrix, where N is the number of grid characters. The \mathbf{Q} matrix can either be the cumulative classifier score rankings from the CMA static stopping algorithm or the Bayesian probabilities prior to character selection depending on the error correction method.

If applicable, the trained ErrP classifier was applied to features extracted from a time window of EEG data after the BCI feedback was presented to the user. The ErrP classifier returns a score vector, $\mathbf{S} = [s_1, s_2, \dots, s_T]$ which was used to calculate

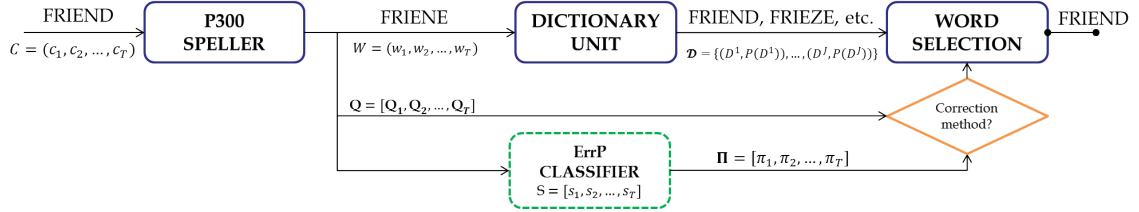


FIGURE 6.6: Flowchart for proposed spelling correction in the P300 speller. A user intends to spell the word C . Using a trained P300 classifier and EEG data, the P300 speller outputs the spelled word, W , and a matrix of character P300 classifier score rankings or probabilities, $\mathbf{Q}^{N \times T}$, (N = number of grid characters, T = length of the spelled word). In the dictionary unit, a list of probable words, \mathcal{D} , based on a string metric function is generated from a vocabulary, with the corresponding prior probabilities, $(D^j, P(D^j))$, obtained from a text corpus. If an ErrP classifier is used, after character selection feedback is presented to the user, the ErrP classifier computes classifier scores, \mathbf{S} , which are used to calculate the ErrP classifier confidences, $\mathbf{\Pi}^{1 \times T}$. Using the word prior probabilities, $P(D^j)$, the \mathbf{Q} matrix or $\mathbf{\Pi}$ vector, the user's intended word choice, \hat{C} , is estimated.

the ErrP classifier confidences, $\mathbf{\Pi} = [\pi_1, \pi_2, \dots, \pi_T]$:

$$\pi_t = \frac{p(s_t|H_c)A_{pr}}{p(s_t|H_c)A_{pr} + p(s_t|H_e)(1 - A_{pr})} \quad (6.3)$$

where π_t is the confidence that the character w_t is correct; $p(s_t|H_c)$ and $p(s_t|H_e)$ are the likelihoods that the ErrP classifier score, s_t , is generated from a correct character and incorrect character (hence ErrP elicited), respectively; A_{pr} is the projected accuracy [53] of the P300 speller selection process calculated from the training data of the first session.

Spelling Correction

Once the outcomes of the P300 speller and ErrP detector, if applicable, were simulated, six methods of error correction were applied and the performances compared. For language-based error correction methods, for a spelled word, W , a set of possible word choices, $\mathcal{D} = \{D^1, D^2, \dots, D^3\}$, were selected from a dictionary using the Levenshtein edit distance [93]. The dictionary vocabulary ($\approx 30,000$ words) was created from a modified corpus compiled by Norvig [112]. Word choices were limited

to those of the same length as W , within a minimum Levenshtein edit distance from the spelled word. Some error correction methods relied on the unigram probabilities of word choices, and so word-probability pairs, $(D^i, P(D^i))$, were generated from the corpus. Frequency counts of the dictionary words were smoothed to obtain word unigram probabilities, $P(D^j)$.

For some algorithms, a noisy channel model was used for spelling correction:

$$P(D^j|W, \mathbf{Q}/\mathbf{\Pi}) \propto P(W|D^j, \mathbf{Q}/\mathbf{\Pi})P(D^j) \quad (6.4)$$

$$\hat{C} = \operatorname{argmax}_{D^j \in \mathcal{D}} P(W|D^j, \mathbf{Q}/\mathbf{\Pi})P(D^j) \quad (6.5)$$

where $P(D^j|W, \mathbf{Q}/\mathbf{\Pi})$ is the posterior probability of the word choice D^j , given the spelled word, W , and the \mathbf{Q} matrix from the P300 classifier or the $\mathbf{\Pi}$ vector from the ErrP classifier; and $P(W|D^j, \mathbf{Q}/\mathbf{\Pi})$ is the likelihood of the spelled word given the word choice, D^j , and $\mathbf{Q}/\mathbf{\Pi}$.

(i) *Simple dictionary*: A simple method for error correction is to select the most frequent word from the candidate word choices as the target word. For the simple dictionary correction, the word with the highest unigram probability in \mathcal{D} , was selected as the target word estimate:

$$\hat{C} = \operatorname{argmax}_{D^j \in \mathcal{D}} P(D^j) \quad (6.6)$$

(ii) *Ahi et al., 2011*: While selection based on maximum word frequency can provide a good estimate of the target word, information about the likelihood of characters can enhance the estimate. Ahi *et al.* [94] used the cumulative P300 classifier scores from the CMA static stopping algorithm (2.1) to rank each word choice to obtain an estimate of the user's intended word [94]. The word with the minimum cost (highest rank) was selected as the target word estimate:

$$r(D^j) = \sum_{t=1}^T q_t^{l(D^j)} \quad (6.7)$$

$$\hat{C} = \arg \min_{D^j \in \mathcal{D}} r(D^j) \quad (6.8)$$

where D_t^j is the t^{th} letter of the word D^j ; $l(D_t^j)$ is the grid label for D_t^j ; and $q_t^l(D_t^j)$ is the rank of the classifier score of D_t^j , obtained from the t^{th} column of the \mathbf{Q} matrix, \mathbf{Q}_t . The \mathbf{Q} matrix for this algorithm consists of the cumulative P300 classifier score rankings of characters in the spelled word.

(iii) *Bayesian: A priori* language information and character likelihoods can be combined probabilistically via the noisy channel model (6.4, 6.5) to obtain the target word estimate. The \mathbf{Q} matrix for this algorithm consists of the final character probabilities obtained from P300 speller Bayesian probabilities:

$$P(W|D^j, \mathbf{Q}) = \left(\prod_{t=1}^T q_t^{l(D_t^j)} \right) P(D^j) \quad (6.9)$$

where $q_t^l(D_t^j)$ is the Bayesian character probability of D_t^j , obtained from the t^{th} column of the \mathbf{Q} matrix, \mathbf{Q}_t .

(iv) *Oracle ErrP Classifier*: The oracle ErrP classifier was used to infer the upper bound on the performance of spelling correction that could be obtained with perfect ErrP classification, i.e. “0” is returned for correctly selected characters and “1” for erroneous characters. The set of words in \mathcal{D} was narrowed to words with substitutions only at erroneous character locations. The word with the highest unigram probability in \mathcal{D} was selected as the target word estimate, according to (6.6).

(v) *Perrin et al., 2012*: In this method, erroneous characters flagged by the ErrP classifier were automatically deleted and substituted with the next most probable character [91]. For this study, the ErrP classifier confidences, $\mathbf{\Pi}$, were compared against the participant P300 speller projected accuracy, A_{pr} [53], calculated from the EEG data of the first P300 speller session. For an ErrP classifier confidence $\pi_t < A_{pr}$, character w_t was deleted and substituted with the character that had the second P300 classifier score rank in \mathbf{Q}_t .

(vi) *ErrP Classifier*: Previous ErrP-based error correction methods in P300 speller systems involved auto-deletion of characters with ErrP detection. However, a language-based spelling correction is proposed by using ErrP classifier confidences within a noisy channel model (6.4, 6.5) to estimate the user’s intended word:

$$P(D^j|W, \mathbf{\Pi}) = \left[\prod_{t=1}^T \pi_t^{\left(\delta_{w_t, D_t^j}\right)} \left(\frac{1 - \pi_t}{N - 1}\right)^{\left(1 - \delta_{w_t, D_t^j}\right)} \right] P(D^j) \quad (6.10)$$

where δ_{w_t, D_t^j} is the Kronecker delta, where $\delta = 1$, when $w_t = D_t^j$ and $\delta = 0$, when $w_t \neq D_t^j$; π_t is the ErrP classifier confidence; and $\frac{1 - \pi_t}{N - 1}$ is the remaining ErrP classifier confidence that is evenly distributed across the remaining $N - 1$ characters in the grid.

6.3.2 Results

The character and word accuracies for the raw P300 speller and with spelling correction were calculated. Participant-specific results are shown in Figure 6.7(a), ordered according to raw P300 speller character accuracy. Figure 6.7(b) and (c) show pooled participant results. Statistical analyses for the character and word accuracies revealed a significant difference in the means of at least two algorithms ($p < 0.05$), and pairwise comparisons are shown in Table 6.1 and 6.2.

Figure 6.7(b) compares character-based correction with the ErrP classifier to word-based correction with a simple dictionary correction. It can be observed that correcting whole words with errors is more beneficial compared to the character-based method proposed by Perrin *et al.* [91], as word-based correction utilises the positional context of errors to generate word alternatives. Even with just one sequence of data prior to character selection, a simple dictionary correction was able to yield a significant increase in participant accuracy, with 35-185% improvement in character accuracy.

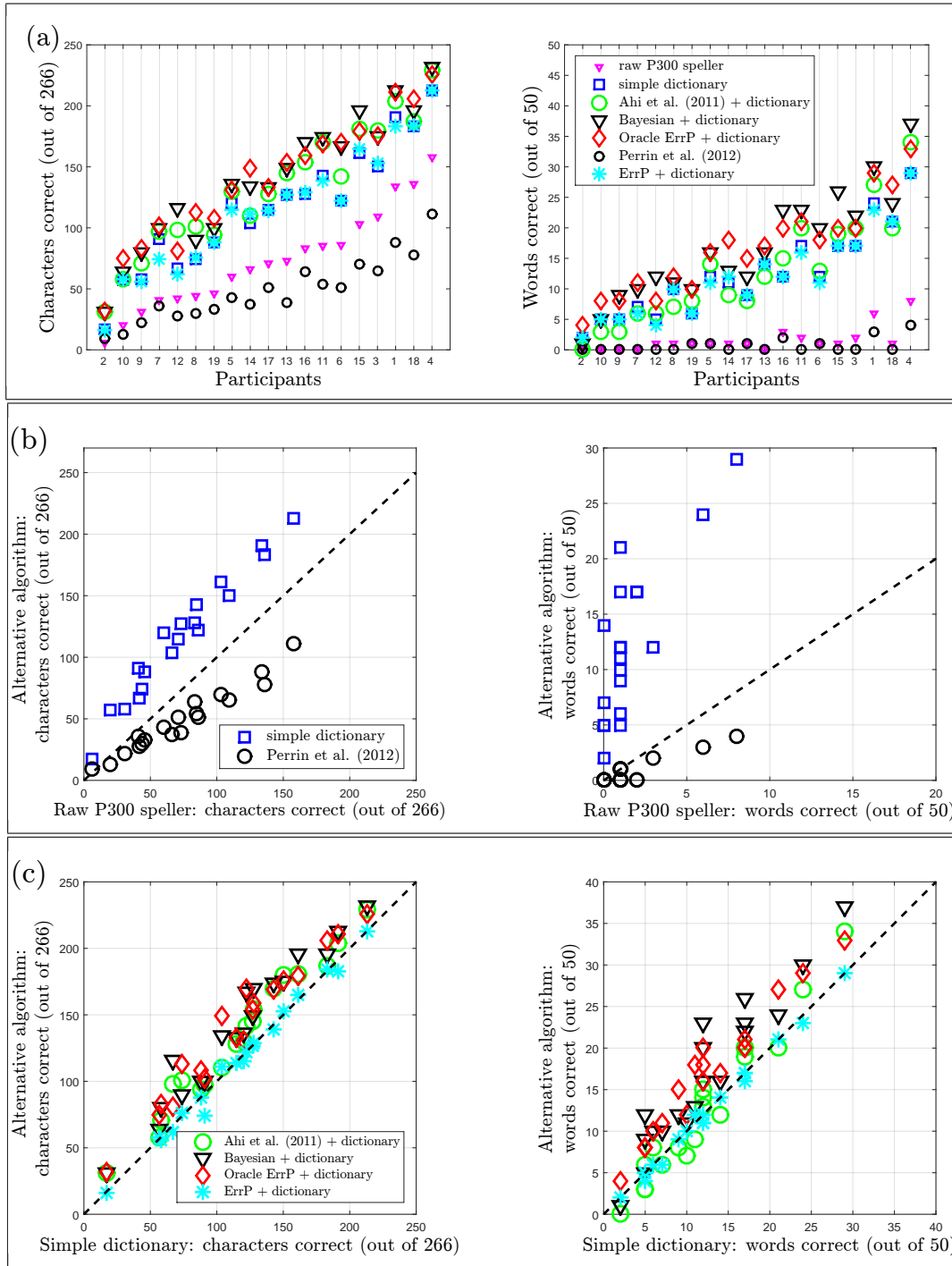


FIGURE 6.7: Participant results for P300 speller accuracy, with and without use of spelling correction algorithms. (a) Participant-specific results for all the correction algorithms, ordered by increasing raw P300 speller character accuracy. (b) Pooled results comparing character-based (Perrin *et al.* [91]) and word-based (simple dictionary) spelling correction. (c) Pooled results comparing the effect of including additional information from either the ErrP or P300 classifier to a simple dictionary spelling correction

Successfully deleting and replacing erroneous characters that are flagged by an ErrP classifier requires high discriminability performance by the ErrP classifier. At these ErrP detection performances, utilising the Perrin *et al.* method of replacing erroneous characters with the next most probable character negatively impacted accuracy, ranging from -51 to 0%. It is possible that the Perrin *et al.* correction method was adversely affected by the limited amount of data collection prior to character selection, as more data could have led to sparser character probability distributions where likely and unlikely characters are better separated. However, there is no guarantee that if the ErrP classifier correctly flags and deletes an erroneous character, substitution with the next most probable character will correspond with the target. These results highlight the benefit of including language information, especially under the challenging condition of limited training data for ErrP classification.

Figure 6.7(c) illustrates the potential benefit that could be obtained by adding information about the confidence in each character to simple dictionary correction. The performance of the ErrP classifier correction method (32-185%) is comparable to that of a simple dictionary correction, suggesting no additional benefit in attempting to correct errors at these ErrP detection rates. However, with perfect ErrP detection, a significant benefit occurs (43-433%), suggesting that the knowledge of incorrect characters can be beneficial to word correction. Relying on cumulative P300 classifier score rankings of letters to rank words, as in Ahi *et al.* [94], has some benefit to word correction (37-416%); however, the Bayesian approach that uses a noisy channel model further improves performance (44-416%). Furthermore, performance with the noisy channel model with Bayesian character probabilities is similar to that with perfect ErrP detection. This suggests that training an additional ErrP classifier to flag erroneous characters in the P300 speller may not be necessary as spelling correction can be achieved with a dictionary by utilizing the error information that is encoded in the P300 classifier responses.

Table 6.1: Statistical pairwise comparison between correction algorithms: Character performance

ALGORITHM	Raw P300 speller	Simple dictionary	Ahi <i>et al.</i> + dictionary	Bayesian + dictionary	Oracle ErrP + dictionary	Perrin <i>et al.</i>	ErrP + dictionary
Mean \pm Std	73.32 \pm 41.11	115.47 \pm 51.14	131.47 \pm 52.80	139.26 \pm 54.10	139.37 \pm 52.80	49.84 \pm 26.88	114.95 \pm 50.19
Raw P300 speller		↓	↓	↓	↓	↑	↓
Simple dictionary			↓	↓	↓	↑	
Ahi <i>et al.</i> + dictionary						↑	↑
Bayesian + dictionary						↑	↑
Oracle ErrP + dictionary						↑	↑
Perrin <i>et al.</i> ErrP							↓
ErrP + dictionary							

Character accuracy is out of 266 characters. Analysis performed using repeated measures ANOVA (p -value < 0.05), with Bonferroni adjustment for pair-wise comparisons.

LEGEND: ↑, significantly higher; ↓, significantly lower. Legend entries are interpreted row-wise.

Example: Entry ↑ in (x, y) means performance of the algorithm in row x is significantly higher than that in column y .

Table 6.2: Statistical pairwise comparison between correction algorithms: Word performance

ALGORITHM	Raw P300 speller	Simple dictionary	Ahi <i>et al.</i> + dictionary	Bayesian + dictionary	Oracle ErrP + dictionary	Perrin <i>et al.</i>	ErrP + dictionary
Mean \pm Std	1.58 \pm 2.09	12.16 \pm 7.34	12.74 \pm 8.90	16.63 \pm 9.25	16.37 \pm 7.92	0.74 \pm 0.93	11.74 \pm 7.24
Raw P300 speller		↓	↓	↓	↓		↓
Simple dictionary				↓	↓	↑	
Ahi <i>et al.</i> + dictionary				↓		↑	
Bayesian + dictionary						↑	↑
Oracle ErrP + dictionary						↑	↑
Perrin <i>et al.</i> ErrP							↓
ErrP + dictionary							

Word accuracy is out of 50 words. Analysis performed using repeated measures ANOVA (p -value < 0.05), with Bonferroni adjustment for pair-wise comparisons.

LEGEND: ↑, significantly higher; ↓, significantly lower. Legend entries are interpreted row-wise.

Example: Entry ↑ in (x, y) means performance of the algorithm in row x is significantly higher than that in column y .

6.4 Discussion

The BCI selection process is error prone and erroneous selections have to be detected and corrected to effectively convey a user’s intent in a timely fashion. Attempting to correct every erroneous selection leads to an increase in spelling time, sometimes to levels that may not be practical for communication. The potential increase in spelling time can be predicted based on modeling the dynamics of the error correction mechanism to evaluate the utility of an error correction algorithm. Theoretical estimates for the effective number of selections required to achieve 100% accuracy, which were verified with simulations of different error correction mechanisms, can be

be used to evaluate the utility of error correction mechanisms.

The conventional method of active user correction, where the user initiates the erroneous character revisions, is limited in efficacy to users with high accuracy levels. ErrP-based correction has been proposed as an alternative error correction mechanism for BCI systems. With a P300-ErrP cascade, a user can potentially save the time necessary to initiate and execute a deletion command with automatic vetoes upon ErrP detection. However, the overall benefit of a hybrid P300-ErrP correction mechanism needs to be evaluated based on the combined performance characteristics of P300 speller selection and ErrP detection.

ErrP detection requires the collection of a substantial amount of training data in order to be accurate, and the accuracy of the detection drives the efficacy of ErrP-based correction mechanisms. From the comparison study, the difference between the accuracies of the trained ErrP-based correction method and the oracle ErrP-based correction method was statistically significant, both for characters (difference of approximately 25 characters correct) and words (difference of approximately 4 words correct). This suggests that additional training data would be required to achieve the full potential of ErrP-based correction. However, by relying on language information and P300 classifier outputs, equivalent performance to the oracle ErrP-based correction method was achieved at a much reduced cost in time and effort.

The Bayesian correction algorithm has the further advantage of being applicable to other ERP-based spelling BCIs with probabilistic data collection algorithms. In addition, it can also be incorporated within any probabilistic-based spelling correction algorithm. Spell-checking algorithms have been widely studied for other applications and can be exploited for BCI spelling applications [117]. For example, although unigram word probabilities were used, additional context within sentences can be provided via higher order n -gram language models for context-based spelling correction, especially for detecting and correcting real-word errors. While an on-line

implementation was beyond the scope of this study, the oracle ErrP-based correction algorithm provided an estimate of the upper bound on performance with an ErrP-based correction system. In this offline analysis, the Bayesian correction method achieved similar performance to the upper bound of an ErrP-based system, suggesting the potential for correction without ErrP detection.

However, the Bayesian spelling algorithm requires further development prior to on-line BCI spelling applications. The performance of dictionary-based correction is dependent on the language models developed from a compiled corpus. A user-specific body of text can provide more language context and it can be updated and smoothed periodically to handle out-of-vocabulary words. Another issue is the detection of word boundaries/space characters prior to performing spelling correction. Most P300 speller studies design their spelling tasks with single words and in this study, the words were extracted from phrases, hence the target word length is known *a priori*. In addition, dictionary-based spelling correction is not applicable to numbers or command options if they are BCI choices. Natural language processing tools like word segmentation/tokenisation [118] or techniques from optical character recognition [119] can be exploited to further improve the performance of BCI spellers for more practical use for the target BCI population.

Conclusions and Future Work

The focus of this research was to develop adaptive algorithms that take into account the physical limitations of the target BCI population to improve the efficiency of ERP-based spellers for real-world communication. Given the non-stationarity of a user's BCI performance, inter-user performance variability and the wide array of messages that can be communicated with a BCI system, it was hypothesised that building adaptive capabilities into the BCI system would provide an efficient way for improving BCI speller performance. In this work, three key areas in the BCI system framework were identified for improving user performance: the visual interface, the target character estimation process and correcting the target selection process. In addition, developing a BCI performance prediction method that is flexible to a wide array of BCI modalities provides a useful cost and time efficient tool to assess BCI performance. The major contributions of this work are summarised below, as well as suggestions for future work.

BCI Performance Prediction

Previous methods for BCI performance prediction that have been presented in the literature rely on regression analysis using noisy empirical estimates or use a proba-

bilistic model approach that is limited to the row-column paradigm and static data collection. In Chapter 3, a new method was proposed to predict the performance of a generic BCI system based on random variable analysis of the BCI target character estimation algorithm. Specifically, for the P300 speller, the proposed method is independent of the stimulus presentation paradigm and accounts for dynamic data collection. The proposed method also accounts for changes in a user's performance level as performance is quantified with the classifier detectability index. Results from previous online studies with the Bayesian dynamic stopping algorithm provided validation for the proposed method as online participant performances were consistent with predictions.

The new prediction method has significant theoretical and practical implications as it provides a powerful offline analysis tool to assess BCI performance without extensive online testing. Although the focus of this work was on P300 spellers, the random variable analysis technique presented here to derive BCI performance measures could be applicable to any decision-based BCI in which 1-of-M choices is selected. A similar methodology could be employed to validate the proposed prediction method in other BCI modalities.

Information Optimisation

The stimulus presentation paradigm of an ERP-based BCI controls how ERPs are elicited in the brain. The relative strengths of the ERP responses are affected by stimulus timing parameters. In particular, the time interval between target stimulus events affects the ability of the brain to elicit a strong P300 ERP response with every target character presentation. These TTI-related effects can limit the performance of a given stimulus paradigm, especially if short TTIs are predominant. It was hypothesised that BCI performance can be improved by maximising the information content that is presented to the user via the stimulus presentation pattern: an approach that exploits coding theory by modeling the BCI as a noisy communication

channel. However, previous approaches to exploit information theoretics in designing new stimulus presentation paradigms did not lead to improved performances as had been predicted. It was hypothesised that these previous approaches failed to properly model the P300 channel dynamics as that of a channel with long term memory to target stimulus events.

In Chapter 4, a new method was proposed that uses performance-based parameters in a greedy search algorithm to develop a codebook for a new stimulus paradigm that minimises TTI-related effects and improves performance. In particular, the proposed stimulus paradigm development method is a theoretical application of the BCI performance prediction method developed in Chapter 3. In a time efficient manner, the prediction method was used to estimate and compare the relative performances of a large number of possible codeword combinations prior to any performance evaluation with simulations or online testing. In an online study, the new performance-based paradigm statistically significantly improved performance across a wide range of participant performance levels from the conventional row-column paradigm. This study also provided further validation of the proposed BCI performance prediction method as the online results were consistent with offline predictions. By considering a more realistic model of the P300 channel dynamics, this work demonstrates that BCI performance can be improved with a principled design of the stimulus presentation paradigm. The new performance-based paradigm requires further testing to confirm these results from non-disabled participants in a target BCI user population.

It is hypothesised that the stimulus presentation paradigm development process can be further adapted with language information. In this work, a uniform distribution was assumed over characters when developing the performance-based paradigm. This work suggests that a language model can be used to guide the character-to-codeword assignment process such that the temporal correlation of the cumulative responses between characters that are highly likely is minimised. Also, the language

model can be used to determine the frequency of presentation such that characters that are highly likely are presented relatively more frequently than other characters that are not as likely. For example, in the case of a bigram language model, a new codebook can be developed based on letter probabilities as determined from the previously selected character.

Target Character Estimation

BCI use requires repeated data measurements to enhance the low SNR associated with brain responses that are used as control signals in order to improve the accuracy of the target character estimation process. The conventional static data collection strategy limits the BCI from adapting to acute changes in a user's performance level during BCI use. Dynamic data collection has been shown to improve performance by adapting the amount of data collection based on a user's current SNR conditions [24]. In this work, a viable Bayesian dynamic stopping algorithm was developed and tested in both healthy and target BCI users, with results indicating the potential advantage of using an adaptive data collection to improve P300 speller efficiency. Importantly, the Bayesian dynamic stopping algorithm was well-received when tested in a target BCI user population.

Additional benefits can be derived with the inclusion of language information in the target character estimation algorithm. It was hypothesised that instead of a simple letter bigram model, a more complex language model that exploits more of the user's spelling history might potentially improve performance in the ALS population. In Chapter 5, an enhanced Bayesian dynamic stopping algorithm that included a more complex language model was proposed, the n -gram DAPS model, which uses all of a user's spelling history. The n -gram DAPS model exploits the Bayesian probabilities obtained from previous character selections to generate multiple word prefix options to develop the language model probabilities for the subsequent character. The language model considers multiple prefix options that are validated with a dic-

tionary, and this potentially minimises the need for erroneous character revisions. In an online study using non-disabled participants, a statistically significant reduction in the character selection time was obtained with the n -gram DAPS model compared to the bigram model. Also, there is the potential to improve accuracy with higher order language models in lower performing participants.

The complexity of the language information utilised in BCI spellers could be further increased to adapt the BCI for more realistic communication scenarios such as free-spelling sentences or phrases. This includes the use of functional commands, such as a “backspace” character, and additional considerations for word-space boundaries. Predictive word options have the potential to increase the BCI speller communication rates, especially for low-performing participants who generally have relatively higher character selection times. However, the manner by which this information is integrated into a BCI speller can negatively affect performance especially if it requires a changing user interface with multiple menu options, e.g. [86]. Nonetheless, a few online studies have shown that the cognitive load of a predictive user interface paradigm can be minimised by using a word selection process that is similar to single-character selection. For example, by either using a two-step word selection process as in [120] or integrating the predictive word options directly into the speller interface, as in [87]. In both studies, a significant increase in spelling speed was obtained with both predictive spellers compared to a conventional speller interface. In Kaufmann *et al.*, (2012) there was no significant detriment to participant accuracy using the predictive speller [87].

Directly integrating predictive word options into the BCI speller grid as in [87] allows the flexibility for a user to either select one of the suggested word options or continue with single-character selection. However, previous methods of integrating word prediction (e.g. [87, 120]) used only word frequency to select the word options. This work suggests that there might be a positive benefit to including additional

information based on the likelihood of characters obtained from the target character estimation process. The word options could be generated based on the prefix search method from the n -gram DAPS model, with the word options updated, if necessary, based on the user’s spelling history. A simpler language model can still be used in the character probability initialisation step, taking into consideration the word options in the grid. It is hypothesised that the dual use of language information in the target character estimation process as well as the user interface will offer more opportunities to increase the BCI speller throughput.

Error Correction

Automatic error identification and correction has the potential to improve BCI communication efficiency. ErrP-based corrective mechanisms that use elicited brain responses to errors have been proposed to automatically delete erroneous BCI actions upon ErrP detection. However, implementing ErrP-based mechanisms comes at a cost in data and time. Rather than perform corrective actions on a character-by-character basis, this work suggests word-based spelling correction as a more effective alternative as it uses *a priori* language information by considering the positional context of errors in words.

In this work, a spelling correction algorithm was proposed within the framework of probabilistic data collection algorithms for BCI spellers that does not rely on ErrP detection: a noisy channel model (NCM) with Bayesian character probabilities which combines information that is already available from the target character estimation process and dictionary-based suggestions to correct misspelled words. The offline analysis presented in Chapter 6 demonstrates that the proposed Bayesian NCM spelling correction algorithm achieves comparable improvements from raw P300 speller accuracy to that of an “oracle” ErrP-based correction method for which perfect ErrP detection was assumed [116]. This suggests that the Bayesian NCM method with P300 classifier confidences may provide a more reliable approach to

spelling correction than developing an ErrP-based correction system. The Bayesian correction algorithm has the further advantage of being applicable to other ERP-based spelling BCIs with probabilistic data collection algorithms. In addition, the Bayesian correction method can also be incorporated within any probabilistic-based spelling correction algorithm. While an on-line implementation was beyond the scope of this study, the oracle ErrP-based correction algorithm provided an estimation of an upper bound for an ErrP-based correction system. In the offline analysis, the Bayesian correction method achieved similar performance to the upper bound of an ErrP-based system, suggesting the potential for error correction without ErrP detection.

However, the Bayesian spelling correction algorithm requires further development prior to online BCI implementation. The performance of dictionary-based correction is dependent on the language models developed from a compiled corpus. A user-specific body of text provides more language context and it can be updated and smoothed periodically to handle out-of-vocabulary words. Although unigram word probabilities were used in this study, additional context within sentences can be provided via higher order language models for context-based spelling correction, especially for detecting and correcting real-word errors [117]. Another issue is the detection of word boundaries or space characters prior to performing spelling correction within the context of phrases or sentences. Most P300 speller studies, including those in this work, design spelling tasks with single word copy-spelling. In this spelling correction study, the words were extracted from phrases, hence the target word length was known *a priori*. In addition, dictionary-based spelling correction is not applicable to numbers or command options in the speller grid. Other natural language processing tools like word segmentation/tokenisation [118] or techniques from optical character recognition [119] can be exploited to further improve the performance of BCI spellers for more practical use for the target BCI population.

In summary, a BCI user is constantly adapting and modifying their behaviour in real-time in response to device use and feedback. Building adaptive capabilities into BCI algorithms can potentially give the BCI system the flexibility to improve P300 performance by adjusting system parameters in response to changing user inputs. The parameters of the stimulus presentation paradigm can be modified to modulate and enhance the elicited ERPs to improve user performance. The predictability of language can be exploited to improve the target character estimation process and to perform error corrections. The studies presented in this work provide evidence that the methods proposed in this document for incorporating adaptive strategies in the stimulus paradigm, target character estimation algorithm and for post-selection error correction, have the potential to improve BCI communication rates, both in accuracy and spelling speed. This research hopes to contribute towards transitioning the BCI from a research tool to a viable communication alternative for daily home use to improve the quality of life of individuals with severe physical limitations by enabling them interact with their family, friends and environment.

Appendix A

Random Sums

Let X and Y , be two continuous random variables, with probability density functions (pdf), $f_X(x)$ and $f_Y(y)$, respectively. Their sum, $Z = X + Y$ is a random variable with pdf $f_Z(z)$, where f_Z is the convolution of f_X and f_Y :

$$\begin{aligned} f_Z(z) &= (f_X * f_Y)(z) \\ &= \int_{-\infty}^{\infty} f_X(z - y)f_Y(y)dy \\ &= \int_{-\infty}^{\infty} f_Y(z - x)f_X(x)dx \end{aligned} \tag{A.1}$$

Hence the pdf of a sum of more than two random variables, $\Omega = X + Y + \dots + Z$, can be obtained via successive convolutions:

$$f_{\Omega}(\omega) = (f_X * f_Y * \dots * f_Z)(\omega) \tag{A.2}$$

$$\tag{A.3}$$

Given a sequence of independent and identically distributed random variables, Z_1, Z_2, \dots, Z_T , consider the random sum [97]:

$$S = Z_1 + Z_2 + \dots + Z_T \quad (\text{A.4})$$

where the number of observations, T is also a random variable. For a fixed value, $T = t$, the conditional expected value of S can be obtained:

$$\begin{aligned} \mathbb{E}[S|T = t] &= \sum_{i=1}^t \mathbb{E}[Z_i] \\ &= t\mathbb{E}[Z] \end{aligned} \quad (\text{A.5})$$

where $\mathbb{E}[\cdot]$ is the expected value of the random variable. Using the theorem of total expectation, the expected value of S can be obtained:

$$\begin{aligned} \mathbb{E}[S] &= \sum_t t\mathbb{E}[Z]p_T(t) \\ &= \mathbb{E}[Z] \sum_t tp_T(t) \\ &= \mathbb{E}[Z]\mathbb{E}[T] \end{aligned} \quad (\text{A.6})$$

(A.6) is referred to as *Wald's equation*.

Appendix B

Random Inequalities

Given two random variables, X and Y , with joint probability density function (pdf), $f_{X,Y}(x, y)$, and marginal pdfs, $f_X(x)$ and $f_Y(y)$, respectively, the probability that X is less than Y can be determined accordingly:

$$\begin{aligned} P(X < Y) &= \int_{-\infty}^{\infty} P(X < Y|Y = t) f_Y(t) dt \\ &= \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^t f_{X|Y=t}(t) \right\} f_Y(t) dt \\ &= \int_{-\infty}^{\infty} F_{X|Y=t}(t) f_Y(t) dt \end{aligned} \tag{B.1}$$

where $F_{X|Y}$ is the conditional cumulative distribution function (cdf) of X given Y . If X and Y are independent, then their marginal pdfs are independent of each other:

$$P(X < Y) = \int_{-\infty}^{\infty} F_X(t) f_Y(t) dt \tag{B.2}$$

For example, for a bivariate normal distribution with mean vector, \mathbf{M} , and covariance matrix Σ ,

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathbf{M} = \begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \mathbf{\Sigma} = \begin{bmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_Y\sigma_X & \sigma_Y^2 \end{bmatrix}$$

the parameters of the conditional pdf, $f_{X|Y}(x|y)$, are:

$$\mu_{X|Y} = \mu_X + \rho \left(\frac{\sigma_X}{\sigma_Y} \right) (Y - \mu_Y) \quad (\text{B.3a})$$

$$\sigma_{X|Y}^2 = \sigma_Y^2 (1 - \rho^2) \quad (\text{B.3b})$$

If $\rho = 0$, i.e. X and Y are uncorrelated, hence independent since they are jointly Gaussian, a closed form solution can be obtained:

$$P(X < Y) = \Phi \left(\frac{\mu_Y - \mu_X}{\sqrt{\sigma_Y^2 + \sigma_X^2}} \right) \quad (\text{B.4})$$

where $\Phi(\cdot)$ is the cdf of the standard normal probability distribution.

Appendix C

The Maximum of M Random Variables

Given M random variables, X_1, \dots, X_M , the probability that the maximum of the random variables, $\max(X_1, \dots, X_M)$, is less than a certain value, z , $P(\max(X_1, \dots, X_M) < z)$ is equivalent to evaluating the joint cumulative distribution function (cdf) of (X_1, \dots, X_M) at $z\mathbf{1}_{1 \times M}$, $P(X_1 < z, \dots, X_M < z)$.

Alternatively, a random variable can be defined, $Z = \max(X_1, \dots, X_M)$, with cdf, $F_Z(z) = P(Z < z)$.

- If the X_i 's are independent, then the cdf of Z is the product of the individual cdfs of the M random variables [97]:

$$F_Z(z) = \prod_{i=1}^M F_i(z) \tag{C.1}$$

- If the X_i 's are correlated and not necessarily identically distributed, some approaches have been proposed in the literature to approximate the probability distribution of $Z = \max(X_1, X_2, \dots, X_M)$ [105, 106, 121, 122, 123].

The most commonly used is Clark's formula for approximating the probability distribution of the maximum of correlated normal random variables [105]. Given two variables, $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, with correlation coefficient, ρ , the probability distribution of $Z = \max(X_1, X_2)$, can be approximated with another Gaussian, $Z \sim \mathcal{N}(\mu_Z, \sigma_Z^2)$:

$$a = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2} \quad (\text{C.2a})$$

$$\alpha = (\mu_1 - \mu_2)/a \quad (\text{C.2b})$$

$$\mu_Z = \mu_1\Phi(\alpha) + \mu_2\Phi(-\alpha) + a\phi(\alpha) \quad (\text{C.2c})$$

$$\sigma_Z^2 = (\mu_1^2 + \sigma_1^2)\Phi(\alpha) + (\mu_2^2 + \sigma_2^2)\Phi(-\alpha) + (\mu_1 + \mu_2)a\phi(\alpha) - \mu_Z^2 \quad (\text{C.2d})$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are probability density function (pdf) and cdf values evaluated using the standard normal distribution. Extending from (C.2a-C.2d), the correlation coefficient of the maximum of two random variables, $\max(X_1, X_2)$, and another random variable, X_3 is:

$$\rho(X_3, \max(X_1, X_2)) = \frac{\sigma_1\rho_{1,3}\Phi(\alpha) + \sigma_2\rho_{2,3}\Phi(-\alpha)}{\sigma_Z} \quad (\text{C.3})$$

where $\rho_{i,j}$ is the correlation coefficient of X_i and X_j .

Hence, given a set of normally distributed random variables, the parameters of the probability distribution of $\max(X_1, X_2, \dots, X_M)$ can be calculated recursively:

$$\begin{aligned} \max(X_1, X_2, \dots, X_M) &= \max(X_M, \max(X_1, \dots, X_{M-1})) \\ &= \max(X_M, \max(X_{M-1}, \max(X_1, \dots, X_{M-2}))) \end{aligned} \quad (\text{C.4})$$

Appendix D

Truncated Sequential Likelihood Ratio Test

The truncated sequential likelihood ratio test (SLRT) is similar to the untruncated SLRT (3.7), but for an imposed data collection limit, t_{\max} and a new decision rule at t_{\max} [103]:

$$\text{For } t < t_{\max}, \Lambda_t \begin{cases} \leq \mathcal{A} \Rightarrow \text{stop, choose } H_0 \\ \geq \mathcal{B} \Rightarrow \text{stop, choose } H_1 \\ \in (\mathcal{A}, \mathcal{B}) \Rightarrow \text{take another sample} \end{cases} \quad (\text{D.1a})$$

$$\text{For } t = t_{\max}, \Lambda_t \begin{cases} < \Gamma \Rightarrow \text{choose } H_0 \\ \geq \Gamma \Rightarrow \text{choose } H_1 \end{cases} \quad (\text{D.1b})$$

where t_{\max} is the pre-set data truncation limit, \mathcal{A} and \mathcal{B} are defined according to (3.7), and Γ is a threshold value to decide on a hypothesis at $t = t_{\max}$ with $A \leq \Gamma \leq B$.

To evaluate the performance of the truncated SLRT conditioned on a hypothesis, H_i , the probability density function of the cumulative sum of the log-likelihood ratios, $\ln \Lambda_t = \sum_{\tau=1}^t \ln \lambda_{\tau}$, has to be evaluated, given that $\mathcal{A} < \ln \Lambda_n < \mathcal{B}$ for $n = 1, \dots, t - 1$. For a given hypothesis, H_i , let $P_i(z, t)$ denote the cdf probability, $P(\ln \Lambda_t \leq$

$z, \mathcal{A} < \ln \Lambda_n < \mathcal{B}, n = 1, \dots, t - 1$), with the corresponding pdf, $p_i(z, t)$. Successive convolutions are required to determine the pdf, $p_i(z, t)$, at different time indices:

$$p_i(z, 1) = g_i(z) \quad (\text{D.2a})$$

$$p_i(z, t) = \int_a^b p_i(u, t-1) g_i(z-u) du, \quad t > 1 \quad (\text{D.2b})$$

where $g_i(z)$ is the pdf of $\ln \lambda$ under a given a hypothesis, H_i , i.e. $p(\ln \lambda | H_i)$; $a = \ln \mathcal{A}$ and $b = \ln \mathcal{B}$.

The performance functions for the truncated SLRT can then be derived:

$$\begin{aligned} \mathbb{E}^{tr}[T|H_i] &= \sum_{t=1}^{t_{\max}} t P(T = t) \\ &= \sum_{t=1}^{t_{\max}-1} t \left\{ \int_{-\infty}^a p_i(z, t) dz + \int_b^{\infty} p_i(z, t) dz \right\} + t_{\max} \int_a^b p_i(z, t_{\max} - 1) dz \end{aligned} \quad (\text{D.3a})$$

$$P("H_0" | H_i) = \sum_{t=1}^{t_{\max}-1} \left\{ \int_{-\infty}^a p_i(z, t) dz \right\} + \int_{-\infty}^{\ln \Gamma} p_i(z, t_{\max}) dz \quad (\text{D.3b})$$

$$P("H_1" | H_i) = \sum_{t=1}^{t_{\max}-1} \left\{ \int_b^{\infty} p_i(z, t) dz \right\} + \int_{\ln \Gamma}^{\infty} p_i(z, t_{\max}) dz \quad (\text{D.3c})$$

where $\mathbb{E}^{tr}[T|H_i]$ is the average number of observations, given that H_i is the true hypothesis; $P("H_j" | H_i)$ is the probability of deciding H_j , given that the true hypothesis is H_i .

Appendix E

Character Probability Initialisation in the Forward Algorithm

Algorithm 6: Forward Algorithm with a Second Order Markov Model [124]

Input: M = Number of model states, $\{s_1, s_2, \dots, s_M\}$; \mathbf{O}_T = observations, (o_1, o_2, o_T) ; Π = initial probability vector; \mathbf{a} = transition probability matrix; \mathbf{b} = state observation likelihood function

Output: α_T = Character initialisation probabilities

- 1 **Function** forwardAlgorithm($M, \mathbf{O}_T, \Pi, \mathbf{a}, \mathbf{b}$)
- 2 \triangleright Probabilities computed $\forall(i, j, k) = 1, 2, \dots, M$
- 3 **for** $t = 1 : T$ **do**
- 4 **if** $t = 1$ **then**
- 5 $\alpha_1(i) = \pi_i b_i(\mathbf{O}_t)$
- 6 **else if** $t=2$ **then**
- 7 $\alpha_2(i, j) = \alpha_1(i) a(i, j) b_j(\mathbf{O}_t)$
- 8 $\alpha_2(j) = \sum_i \alpha_2(i, j)$
- 9 **else**
- 10 $\alpha_t(j, k) = \sum_i \alpha_{t-1}(i, j) a(i, j, k) b_k(\mathbf{O}_t), 3 \leq t \leq T$
- 11 $\alpha_t(k) = \sum_j \alpha_t(j, k), 3 \leq t \leq T$
- 12 **end**
- 13 **end**
- 14 **return** α_T

(c_1, c_2, \dots, c_T) = A sequence of states

Initial probabilities: $\pi_i = P(c_1 = s_i)$

Transition probabilities: $a(i, j) = P(c_t = s_j | c_{t-1} = s_i), a(i, j, k) = P(c_t = s_k | c_{t-2} = s_i, c_{t-1} = s_j)$

Emission probabilities: $b_i(\mathbf{O}_t) = P(\mathbf{O}_t | c_t = s_i)$

Forward probabilities: $\alpha_t(i) = P(c_t = s_i, \mathbf{O}_t), \alpha_t(i, j) = P(c_{t-1} = s_i, c_t = s_j, \mathbf{O}_t)$

Bibliography

- [1] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, “Brain-computer interfaces for communication and control,” *Clin Neurophysiol*, vol. 113, no. 6, pp. 767–791, 2002.
- [2] R. Leeb, D. Friedman, G. R. Mller-Putz, R. Scherer, M. Slater, and G. Pfurtscheller, “Self-paced (asynchronous) bci control of a wheelchair in virtual environments: A case study with a tetraplegic,” *Comput Intell Neurosci*, 2007.
- [3] D. J. McFarland, D. J. Krusienski, W. A. Sarnacki, and J. R. Wolpaw, “Emulation of computer mouse control with a noninvasive brain-computer interface,” *J Neural Eng*, vol. 5, no. 2, pp. 101–10, 2008.
- [4] E. K. Chadwick, D. Blana, J. D. Simeral, J. Lambrecht, S. P. Kim, A. S. Cornwell, D. M. Taylor, L. R. Hochberg, J. P. Donoghue, and R. F. Kirsch, “Continuous neuronal ensemble control of simulated arm reaching by a human with tetraplegia,” *J Neural Eng*, vol. 8, no. 3, 2011.
- [5] J. R. Wolpaw and C. B. Boulay, *Brain signals for braincomputer interfaces*. Springer, 2010, pp. 29–46.
- [6] “Amyotrophic lateral sclerosis (als) fact sheet,” *National Institutes of Health*, 2013.
- [7] P. Cipresso, L. Carelli, F. Solca, D. Meazzi, P. Meriggi, B. Poletti, D. Lule, A. C. Ludolph, V. Silani, and G. Riva, “The use of p300-based bcis in amyotrophic lateral sclerosis: from augmentative and alternative communication to cognitive assessment,” *Brain Behav*, vol. 2, no. 4, pp. 479–98, 2012.
- [8] S. Moghimi, A. Kushki, A. M. Guerguerian, and T. Chau, “A review of eeg-based brain-computer interfaces as access pathways for individuals with severe disabilities,” *Assist Technol*, vol. 25, no. 2, pp. 99–110, 2013.

- [9] E. W. Sellers, D. B. Ryan, and C. K. Hauser, “Noninvasive brain-computer interface enables communication after brainstem stroke,” *Science Translational Medicine*, vol. 6, no. 257, p. 257re7, 2014.
- [10] S. Sutton, M. Braren, J. Zubin, and E. R. John, “Evoked-potential correlates of stimulus uncertainty,” *Science*, vol. 150, no. 3700, pp. 1187–8, 1965.
- [11] E. W. Sellers and E. Donchin, “A p300-based brain-computer interface: Initial tests by als patients,” *Clinical Neurophysiology*, vol. 117, no. 3, pp. 538–548, 2006.
- [12] F. Nijboer, E. W. Sellers, J. Mellinger, M. A. Jordan, T. Matuz, A. Furdea, S. Halder, U. Mochty, D. J. Krusienski, T. M. Vaughan, J. R. Wolpaw, N. Birbaumer, and A. Kbler, “A p300-based braincomputer interface for people with amyotrophic lateral sclerosis,” *Clinical Neurophysiology*, vol. 119, no. 8, pp. 1909–1916, 2008.
- [13] E. W. Sellers, T. M. Vaughan, and J. R. Wolpaw, “A brain-computer interface for long-term independent home use,” *Amyotrophic Lateral Sclerosis*, vol. 11, no. 5, pp. 449–455, 2010.
- [14] S. Silvoni, C. Volpato, M. Cavinato, M. Marchetti, K. Priftis, A. Merico, P. Tonin, K. Koutsikos, F. Beverina, and F. Piccione, “P300-based brain-computer interface communication: Evaluation and follow-up in amyotrophic lateral sclerosis,” *Front Neurosci*, vol. 3, p. 60, 2009.
- [15] T. W. Berger, *Brain-Computer Interfaces: An international assessment of research and development trends*. Springer, 2008.
- [16] T. M. Vaughan, D. J. McFarland, G. Schalk, W. A. Sarnacki, D. J. Krusienski, E. W. Sellers, and J. R. Wolpaw, “The wadsworth bci research and development program: At home with bci,” *Ieee Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 229–233, 2006.
- [17] “intendix by g.tec, world’s first personal bci speller,” <http://www.intendix.com/>.
- [18] J. N. Mak, Y. Arbel, J. W. Minett, L. M. McCane, B. Yuksel, D. Ryan, D. Thompson, L. Bianchi, and D. Erdogmus, “Optimizing the p300-based braincomputer interface: current status, limitations and future directions,” *Journal of Neural Engineering*, vol. 8, no. 2, p. 025003, 2011.

- [19] H.-J. Hwang, S. Kim, S. Choi, and C.-H. Im, “Eeg-based brain-computer interfaces: A thorough literature survey,” *International Journal of Human-Computer Interaction*, vol. 29, no. 12, pp. 814–826, 2013.
- [20] A. Kübler, E. Holz, and T. Kaufmann, *Bringing BCI Controlled Devices to End-Users: A User Centred Approach and Evaluation*, ser. Biosystems & Biorobotics. Springer Berlin Heidelberg, 2013, vol. 1, ch. 212, pp. 1271–1274.
- [21] L. A. Farwell and E. Donchin, “Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials,” *Electroencephalogr Clin Neurophysiol*, vol. 70, no. 6, pp. 510–523, 1988.
- [22] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
- [23] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.
- [24] M. Schreuder, J. Hhne, B. Blankertz, S. Haufe, T. Dickhaus, and M. Tangermann, “Optimizing event-related potential based braincomputer interfaces: a systematic evaluation of dynamic stopping methods,” *Journal of Neural Engineering*, vol. 10, no. 3, 2013.
- [25] C. S. Throckmorton, K. A. Colwell, D. B. Ryan, E. W. Sellers, and L. M. Collins, “Bayesian approach to dynamically controlling data collection in p300 spellers,” *IEEE Trans Neural Syst Rehabil Eng*, vol. 21, no. 3, pp. 508–17, 2013.
- [26] R. Chavarriaga, A. Sobolewski, and J. d. R. Milln, “Errare machinale est: The use of error-related potentials in brain-machine interfaces,” *Frontiers in Neuroscience*, vol. 8, no. 208, 2014.
- [27] B. Dal Seno, M. Matteucci, and L. Mainardi, “Online detection of p300 and error potentials in a bci speller,” *Comput Intell Neurosci*, vol. 2010, p. 11, 2010.
- [28] N. M. Schmidt, B. Blankertz, and M. S. Treder, “Online detection of error-related potentials boosts the performance of mental typewriters,” *BMC Neurosci*, vol. 13, p. 19, 2012.
- [29] J. Polich, “Updating p300: an integrative theory of p3a and p3b,” *Clin Neurophysiol*, vol. 118, no. 10, pp. 2128–48, 2007.

- [30] D. J. Krusienski and J. J. Shih, “Control of a visual keyboard using an electrocorticographic brain–computer interface,” *Neurorehabilitation and neural repair*, vol. 25, no. 4, pp. 323–331, 2011.
- [31] P. Brunner, A. L. Ritaccio, J. F. Emrich, H. Bischof, and G. Schalk, “Rapid communication with a p300 matrix speller using electrocorticographic signals (ecog),” *Frontiers in neuroscience*, vol. 5, p. 5, 2011.
- [32] M. Hirata and T. Yoshimine, “Electrocorticographic brain–machine interfaces for motor and communication control,” in *Clinical Systems Neuroscience*, 2015, pp. 83–100.
- [33] D. J. Krusienski, E. W. Sellers, D. J. McFarland, T. M. Vaughan, and J. R. Wolpaw, “Toward enhanced p300 speller performance,” *J Neurosci Methods*, vol. 167, no. 1, pp. 15–21, 2008.
- [34] D. J. Krusienski, E. W. Sellers, F. Cabestaing, S. Bayouhd, D. J. McFarland, T. M. Vaughan, and J. R. Wolpaw, “A comparison of classification techniques for the p300 speller,” *J Neural Eng*, vol. 3, no. 4, pp. 299–305, 2006.
- [35] B. Blankertz, S. Lemm, M. Treder, S. Haufe, and K.-R. Müller, “Single-trial analysis and classification of erp components - a tutorial,” *NeuroImage*, vol. 56, no. 2, pp. 814–825, 2011.
- [36] D. G. Childers, N. W. Perry, I. A. Fischler, T. Boaz, and A. A. Arroyo, “Event-related potentials: a critical review of methods for single-trial detection.” *Critical reviews in biomedical engineering*, vol. 14, no. 3, pp. 185–200, 1986.
- [37] K. M. Spencer, “10 averaging, detection, and classification of single-trial erps,” *Event-related potentials: A methods handbook*, p. 209, 2005.
- [38] M. Billinger, I. Daly, V. Kaiser, J. Jin, B. Z. Allison, G. R. Müller-Putz, and C. Brunner, “Is it significant? guidelines for reporting bci performance,” in *Towards Practical Brain-Computer Interfaces*. Springer, 2013, pp. 333–354.
- [39] D. E. Thompson, S. Blain-Moraes, and J. Huggins, “Performance assessment in brain-computer interface-based augmentative and alternative communication,” *BioMedical Engineering OnLine*, vol. 12, no. 1, p. 43, 2013.
- [40] D. E. Thompson, L. R. Quitadamo, L. Mainardi, S. Gao, P.-J. Kindermans, J. D. Simeral, R. Fazel-Rezai, M. Matteucci, T. H. Falk, and L. Bianchi, “Performance measurement for brain-computer or brain-machine interfaces: a tutorial,” *Journal of neural engineering*, vol. 11, no. 3, p. 035001, 2014.

- [41] B. Blankertz, K. Muller, G. Curio, T. Vaughan, G. Schalk, J. Wolpaw, A. Schlogl, C. Neuper, G. Pfurtscheller, T. Hinterberger, M. Schroder, and N. Birbaumer, “The bci competition 2003: progress and perspectives in detection and discrimination of eeg single trials,” *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 6, pp. 1044–1051, 2004.
- [42] D. J. McFarland, W. A. Sarnacki, and J. R. Wolpaw, “Brain-computer interface (bci) operation: Optimizing information transfer rates,” *Biological Psychology*, vol. 63, no. 3, pp. 237–251, 2003, article.
- [43] S. Kay, *Fundamentals of Statistical Signal Processing, Volume I: Estimation theory*. PTR Prentice-Hall, 1998.
- [44] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume II: Detection Theory*. Englewood Cliffs, N.J.: Prentice-Hall PTR, 1993.
- [45] S. Kay, *Fundamentals of Statistical Signal Processing, Volume III: Practical algorithm development*, ser. Fundamentals of Statistical Signal Processing. Prentice-Hall PTR, 2013.
- [46] D. E. Thompson, S. Warschausky, and J. E. Huggins, “Classifier-based latency estimation: a novel way to estimate and predict bci accuracy,” *Journal of Neural Engineering*, vol. 10, no. 1, p. 016006, 2013.
- [47] A. Bamdadian, C. Guan, K. K. Ang, and J. Xu, “The predictive role of pre-cue eeg rhythms on mi-based bci classification performance,” *Journal of neuroscience methods*, vol. 235, pp. 138–144, 2014.
- [48] A. Lenhardt, M. Kaper, and H. J. Ritter, “An adaptive p300-based online brain computer interface,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 16, no. 2, pp. 121–130, 2008.
- [49] E. Thomas, M. Clerc, A. Carpentier, E. Daucea, D. Devlaminck, and R. Munos, “Optimizing p300-speller sequences by rip-ping groups apart,” in *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, 2013, pp. 1062–1065.
- [50] G. Townsend, B. K. LaPallo, C. B. Boulay, D. J. Krusienski, G. E. Frye, C. K. Hauser, N. E. Schwartz, T. M. Vaughan, J. R. Wolpaw, and E. W. Sellers, “A novel p300-based brain-computer interface stimulus presentation paradigm: moving beyond rows and columns,” *Clin Neurophysiol*, vol. 121, no. 7, pp. 1109–20, 2010.

- [51] J. S. Barlow, “Methods of analysis of nonstationary eegs, with emphasis on segmentation techniques: a comparative review.” *Journal of Clinical Neurophysiology*, vol. 2, no. 3, pp. 267–304, 1985.
- [52] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.
- [53] K. Colwell, C. S. Throckmorton, L. M. Collins, and K. D. Morton, “Projected accuracy metric for the p300 speller,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 22, no. 5, pp. 921–925, 2014.
- [54] T. G. Birdsall, “The theory of signal detectability: Roc curves and their character,” 1966.
- [55] A. J. Simpson and M. J. Fitter, “What is the best index of detectability?” *Psychological Bulletin*, vol. 80, no. 6, pp. 481–488, 1973.
- [56] L. Citi, R. Poli, and C. Cinel, “Exploiting p300 amplitude variations can improve classification accuracy in donchin’s bci speller,” in *Neural Engineering, 2009. NER’09. 4th International IEEE/EMBS Conference on*, 2009, pp. 478–481.
- [57] J. Lu, W. Speier, X. Hu, and N. Pouratian, “The effects of stimulus timing features on p300 speller performance,” *Clinical Neurophysiology*, vol. 124, no. 2, pp. 306–314, 2013.
- [58] S. M. M. Martens, N. J. Hill, J. Farquhar, and B. Schölkopf, “Overlap and refractory effects in a braincomputer interface speller based on the visual p300 event-related potential,” *Journal of Neural Engineering*, vol. 6, no. 2, 2009.
- [59] B. O. Mainsah, K. A. Colwell, L. M. Collins, and C. S. Throckmorton, “Utilizing a language model to improve online dynamic data collection in p300 spellers,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 22, no. 4, pp. 837–846, 2014.
- [60] U. Orhan, K. Hild, D. Erdogmus, B. Roark, B. Oken, and M. Fried-Oken, “Rsvp keyboard: An eeg based typing interface,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012, pp. 645–648.
- [61] R. Fazel-Rezai, “Human error in p300 speller paradigm for brain-computer interface,” *Conf Proc IEEE Eng Med Biol Soc*, vol. 2007, pp. 2516–9, 2007.

- [62] S. Gavett, Z. Wygant, S. Amiri, and R. Fazel-Rezai, “Reducing human error in p300 speller paradigm for brain-computer interface,” in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 2012, pp. 2869–2872.
- [63] M. S. Treder, N. M. Schmidt, and B. Blankertz, “Gaze-independent braincomputer interfaces based on covert attention and feature attention,” *Journal of Neural Engineering*, vol. 8, no. 6, 2011.
- [64] T. Kaufmann, S. M. Schulz, C. Grünzinger, and A. Kübler, “Flashing characters with famous faces improves erp-based braincomputer interface performance,” *Journal of Neural Engineering*, vol. 8, no. 5, p. 056016, 2011.
- [65] Q. Li, S. Liu, J. Li, and O. Bai, “Use of a green familiar faces paradigm improves p300-speller brain-computer interface performance,” *PloS one*, vol. 10, no. 6, p. e0130325, 2015.
- [66] J. Jin, E. W. Sellers, and X. Wang, “Targeting an efficient target-to-target interval for p300 speller braincomputer interfaces,” *Medical & Biological Engineering & Computing*, vol. 50, no. 3, pp. 289–296, 2012.
- [67] C. Polprasert, P. Kukieattikool, T. Demeechai, J. A. Ritcey, and S. Siwamogsatham, “New stimulation pattern design to improve p300-based matrix speller performance at high flash rate,” *Journal of Neural Engineering*, vol. 10, no. 3, p. 036012, 2013.
- [68] R. Hamming, “Error detecting and error correcting codes,” *Bell System Technical Journal*, *The*, vol. 29, no. 2, pp. 147–160, 1950.
- [69] S. Ling and C. Xing, *Coding theory: a first course*. Cambridge University Press, 2004.
- [70] S. Martens, J. M. Mooij, N. J. Hill, J. Farquhar, and B. Schölkopf, “A graphical model framework for decoding in the visual erp-based bci speller,” *Neural computation*, vol. 23, no. 1, pp. 160–182, 2011.
- [71] J. Hill, J. Farquhar, S. Martens, F. Biessmann, and B. Schölkopf, “Effects of stimulus type and of error-correcting code design on bci speller performance,” in *Advances in Neural Information Processing Systems 21*. Curran Associates, Inc., 2009, pp. 665–672.
- [72] J. Geuze, J. D. Farquhar, and P. Desain, “Dense codes at high speeds: varying stimulus properties to improve visual speller performance,” *Journal of neural engineering*, vol. 9, no. 1, p. 016009, 2012.

- [73] E. Lin and J. Polich, “P300 habituation patterns: individual differences from ultradian rhythms,” *Percept Mot Skills*, vol. 88, no. 3 Pt 2, pp. 1111–1125, 1999.
- [74] S. C. Kleih, F. Nijboer, S. Halder, and A. Kübler, “Motivation modulates the p300 amplitude during brain-computer interface use,” *Clin Neurophysiol*, vol. 121, no. 7, pp. 1023–31, 2010.
- [75] F. Nijboer, N. Birbaumer, and A. Kübler, “The influence of psychological state and motivation on brain-computer interface performance in patients with amyotrophic lateral sclerosis - a longitudinal study,” *Front Neurosci*, vol. 4, 2010.
- [76] H. Serby, E. Yom-Tov, and G. F. Inbar, “An improved p300-based brain-computer interface,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 13, no. 1, pp. 89–98, 2005.
- [77] T. Liu, L. Goldberg, S. Gao, and B. Hong, “An online braincomputer interface using non-flashing visual evoked potentials,” *Journal of Neural Engineering*, vol. 7, no. 3, 2010.
- [78] H. Zhang, C. Guan, and C. Wang, “Asynchronous p300-based brain-computer interfaces: A computational approach with statistical models,” *Biomedical Engineering, IEEE Transactions on*, vol. 55, no. 6, pp. 1754–1763, 2008.
- [79] J. Höhne, M. Schreuder, B. Blankertz, and M. Tangermann, “Two-dimensional auditory p300 speller with predictive text system,” in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 2010, pp. 4185–4188.
- [80] J. Park and K. Kim, “A pomdp approach to optimizing p300 speller bci paradigm,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 20, no. 4, pp. 584–594, 2012.
- [81] W. Speier, C. Arnold, J. Lu, A. Deshpande, and N. Pouratian, “Integrating language information with a hidden markov model to improve communication rate in the p300 speller,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 22, no. 3, pp. 678–684, 2014.
- [82] M. Schreuder, J. Hohne, M. Treder, B. Blankertz, and M. Tangermann, “Performance optimization of erp-based bcis using dynamic stopping,” in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, 2011, pp. 4580–4583.

- [83] D. Jurafsky and J. H. Martin, *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*, 2nd ed., ser. Prentice Hall series in artificial intelligence. Upper Saddle River, N.J.: Pearson Prentice Hall, 2009.
- [84] “The cmu pronouncing dictionary,” <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>.
- [85] B. Blankertz, M. Krauledat, G. Dornhege, J. Williamson, R. Murray-Smith, and K.-R. Müller, *A Note on Brain Actuated Spelling with the Berlin Brain-Computer Interface*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4555, ch. 83, pp. 759–768.
- [86] D. B. Ryan, G. E. Frye, G. Townsend, D. R. Berry, G. S. Mesa, N. A. Gates, and E. W. Sellers, “Predictive spelling with a p300-based brain-computer interface: Increasing the rate of communication,” *Int J Hum Comput Interact*, vol. 27, no. 1, pp. 69–84, 2011.
- [87] T. Kaufmann, S. Volker, L. Gunesch, and A. Kübler, “Spelling is just a click away - a user-centered brain-computer interface including auto-calibration and predictive text entry,” *Front Neurosci*, vol. 6, p. 72, 2012.
- [88] B. O. Mainsah, L. M. Collins, K. A. Colwell, E. W. Sellers, D. B. Ryan, K. Caves, and C. S. Throckmorton, “Increasing bci communication rates with dynamic stopping towards more practical use: an als study,” *Journal of Neural Engineering*, vol. 12, no. 1, p. 016013, 2015.
- [89] M. Falkenstein, J. Hohnsbein, J. Hoormann, and L. Blanke, “Effects of cross-modal divided attention on late erp components .2. error processing in choice reaction tasks,” *Electroencephalography and Clinical Neurophysiology*, vol. 78, no. 6, pp. 447–455, 1991.
- [90] A. Buttfield, P. Ferrez, and J. Millan, “Towards a robust bci: error potentials and online learning,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 14, no. 2, pp. 164–168, June 2006.
- [91] P. Margaux, M. Emmanuel, D. Sebastien, B. Olivier, and M. Jeremie, “Objective and subjective evaluation of online error correction during p300-based spelling,” *Advances in Human-Computer Interaction*, vol. 2012, p. 13, 2012.
- [92] M. Spüler, M. Bensch, S. Kleih, W. Rosenstiel, M. Bogdan, and A. Kübler, “Online use of error-related potentials in healthy users and people with severe

- motor impairment increases performance of a p300-bci,” *Clinical Neurophysiology*, vol. 123, no. 7, pp. 1328–1337, 2012.
- [93] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions and reversals,” *Cybernetics and Control Theory*, vol. 10, no. 8, pp. 707–710, 1966.
- [94] S. T. Ahi, H. Kambara, and Y. Koike, “A dictionary-driven p300 speller with a modified interface,” *IEEE Trans Neural Syst Rehabil Eng*, vol. 19, no. 1, pp. 6–14, 2011.
- [95] M. D. Kernighan, K. W. Church, and W. A. Gale, “A spelling correction program based on a noisy channel model,” in *Proceedings of the 13th conference on Computational linguistics*, vol. 2.
- [96] A. Wald, “Sequential tests of statistical hypotheses,” *Ann. Math. Statist.*, vol. 16, no. 2, pp. 117–186, 1945.
- [97] S. Trivedi, *Probability & statistics with reliability, queuing and computer science applications*, 2008.
- [98] G. Albert, “On the computation of the sampling characteristics of a general class of sequential decision problems,” *The Annals of Mathematical Statistics*, pp. 340–356, 1954.
- [99] M. Sobel and A. Wald, “A sequential decision procedure for choosing one of three hypotheses concerning the unknown mean of a normal distribution,” *Ann. Math. Statist.*, vol. 20, no. 4, pp. 502–522, 1949.
- [100] P. Armitage, “Sequential analysis with more than two alternative hypotheses, and its relation to discriminant function analysis,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 12, no. 1, pp. 137–144, 1950.
- [101] B. Eisenberg, “Multihypothesis problems,” in *Handbook of sequential analysis*, B. K. Ghosh and P. K. Sen, Eds., 1991, pp. 229–244.
- [102] C. W. Baum and V. V. Veeravalli, “A sequential procedure for multihypothesis testing,” *Information Theory, IEEE Transactions on*, vol. 40, no. 6, pp. 1994–2007, 1994.
- [103] S. Tantaratana and J. B. Thomas, “Truncated sequential probability ratio test,” *Information Sciences*, vol. 13, no. 3, pp. 283–300, 1977.

- [104] E. Thomas, E. Daucé, D. Devlaminck, L. Mahé, A. Carpentier, R. Munos, M. Perrin, E. Maby, J. Mattout, T. Papadopoulo *et al.*, “Coadapt p300 speller: optimized flashing sequences and online learning,” in *6th International Brain Computer Interface Conference*, 2014.
- [105] C. E. Clark, “The greatest of a finite set of random variables,” *Operations Research*, vol. 9, no. 2, pp. 145–162, 1961.
- [106] R. B. Arellano-Valle and M. G. Genton, “On the exact distribution of the maximum of absolutely continuous dependent random variables,” *Statistics & Probability Letters*, vol. 78, no. 1, pp. 27 – 35, 2008.
- [107] T. Verhoeven, P. Buteneers, J. Wiersema, J. Dambre, and P. Kindermans, “Towards a symbiotic brain–computer interface: exploring the application–decoder interaction,” *Journal of Neural Engineering*, vol. 12, no. 6, p. 066027, 2015.
- [108] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [109] “The english lexicon project.”
- [110] A. Mora-Cortes, N. V. Manyakov, N. Chumerin, and M. M. Van Hulle, “Language model applications to spelling with brain-computer interfaces,” *Sensors*, vol. 14, no. 4, pp. 5967–5993, 2014.
- [111] W. Speier, C. Arnold, J. Lu, R. K. Taira, and N. Pouratian, “Natural language processing with dynamic classification improves p300 speller accuracy and bit rate,” *J Neural Eng*, vol. 9, no. 1, p. 016004, 2012.
- [112] P. Norvig, “How to write a spelling corrector,” <http://norvig.com/spell-correct.html>, 2007.
- [113] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260–269, 1967.
- [114] B. Roark and R. W. Sproat, *Computational approaches to morphology and syntax*. Oxford University Press Oxford, 2007, no. 4.
- [115] B. Mainsah, K. Morton, L. Collins, and C. Throckmorton, “Extending language modeling to improve dynamic data collection in erp-based spellers,” in *6th International Brain-Computer Interface Conference*.

- [116] B. Mainsah, K. Morton, L. Collins, E. Sellers, and C. Throckmorton, “Moving away from error-related potentials to achieve spelling correction in p300 spellers,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 23, no. 5, pp. 737–743, 2015.
- [117] K. Kukich, “Techniques for automatically correcting words in text,” *ACM Comput. Surv.*, vol. 24, no. 4, pp. 377–439, 1992.
- [118] D. D. Palmer, *Tokenisation and sentence segmentation*. Marcel Dekker, Inc., New York, USA, 2000.
- [119] S. Mori, H. Nishida, and H. Yamada, *Optical character recognition*. John Wiley & Sons, Inc., 1999.
- [120] F. Akram, H.-S. Han, H. J. Jeon, K. Park, S.-H. Park, J. Cho, and T.-S. Kim, “An efficient words typing p300-bci system using a modified t9 interface and random forest classifier,” in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, 2013, pp. 2251–2254.
- [121] D. Sinha, H. Zhou, and V. Shenoy, “Advances in computation of the maximum of a set of random variables,” in *Proceedings of the 7th International Symposium on Quality Electronic Design*, 2006, pp. 306–311.
- [122] S. Nadarajah and S. Kotz, “Exact distribution of the max/min of two gaussian random variables,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 210–212, 2008.
- [123] Y. Shi, J. Xiong, V. Zolotov, and C. Visweswariah, “Order statistics for correlated random variables and its application to at-speed testing,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 3, pp. 42:1–42:20, 2013.
- [124] B. Watson and C. Tsoi, “Second order hidden markov models for speech recognition.”

Biography

Boyla Octavie Mainsah was born in the Republic of Cameroon. She received the combined Bachelor of Science and Master of Engineering degrees in Biomedical Engineering from Worcester Polytechnic Institute (WPI), Worcester MA, USA, in 2008. She was the recipient of the WPI Presidential and International Student Scholarships in 2004-2008. She continued her education at Duke University, Durham, NC, USA, under the research supervision of Dr. Leslie Collins and Dr. Chandra Throckmorton. She received the Master of Science degree in Electrical and Computer Engineering from Duke University in May of 2014. She was the recipient of the Kristina M. Johnson Fellowship in 2014-2015. She received the Doctorate of Philosophy degree in Electrical and Computer Engineering from Duke University in March of 2016.

She has authored the following journal publications:

1. Mainsah *et al.*, “Increasing BCI communication rates with dynamic stopping towards more practical use: an ALS study,” *Journal of Neural Engineering*, vol. 12, no. 1, p. 016013, 2015.
2. Mainsah *et al.*, “Moving away from error-related potentials to achieve spelling correction in P300 spellers,” *IEEE Trans Neural Syst Rehabil Eng*, vol. 23, no. 5, pp. 737-743, 2015.
3. Mainsah *et al.*, “Utilizing a language model to improve online dynamic data collection in P300 spellers,” *IEEE Trans Neural Syst Rehabil Eng*, vol. 99, no. 4, pp. 837-746, 2014.

She authored the following book chapter:

1. Mainsah *et al.*, “ALS population assessment of a dynamic stopping algorithm implementation for P300 spellers, In *Brain-Computer Interface Research: A State-of-the-Art Summary*, Springer International Publishing, pp. 79-87, 2015.

She has (co-)authored the following conference abstracts and presentations:

1. Mainsah *et al.*, "Predicting BCI performance with the detectability index," *6th International Brain-Computer Interface Conference*, Asilomar, CA, USA, 2016.
2. Mainsah *et al.*, "Improving P300 speller communication efficiency with dynamic data collection: A study in individuals with ALS," Presentation at the *2nd International Conference on Basic and Clinical Multimodal Imaging*, Utrecht, The Netherlands, 2015.
3. Mainsah *et al.*, "Extending language modeling to improve dynamic data collection in ERP-based spellers," *6th International Brain-Computer Interface Conference*, Graz, Austria, 2014.
4. Mainsah *et al.*, "Enhancing P300 speller performance using dynamic data collection with a language model: An ALS study," *Society for Neuroscience*, San Diego, CA, USA, 2013
5. Mainsah *et al.*, "Achieving spelling correction in P300 Spellers without error-related potentials," *Society for Neuroscience*, San Diego, CA, USA, 2013
6. Collins *et al.*, "Statistical methods for improving brain-computer interface P300 speller speed and accuracy in disabled and non-disabled population," Presentation at the *1st International Conference on Basic and Clinical Multimodal Imaging*, Geneva, Switzerland, 2013.
7. Mainsah *et al.*, "Improving dynamic data collection in P300 spellers with a language model," *5th International Brain-Computer Interface Conference*, Asilomar, CA, USA, 2013.
8. Mainsah *et al.*, "Utilizing a language model to improve dynamic data collection in P300 spellers," *Society for Neuroscience*, New Orleans, LA, USA, 2012.
9. Mainsah *et al.*, "Accelerating dynamic data collection in P300 spellers with language modeling," *EEG and Clinical Neuroscience Society*, Johnson City, TN, USA, 2012.