DESIGN, IMPLEMENTATION AND EVALUATION OF A REAL-TIME P300-BASED BRAIN-COMPUTER INTERFACE SYSTEM

by

ARMAGAN AMCALAR

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University

February 2010

DESIGN, IMPLEMENTATION AND EVALUATION OF
A REAL-TIME P300-BASED
BRAIN-COMPUTER INTERFACE SYSTEM

APPROVED BY:

Assist. Prof. Dr. Müjdat Çetin ……
(Thesis Supervisor)

Assist. Prof. Dr. Ayhan Bozkurt ……

Assoc. Prof. Dr. Albert Levi ……

Assoc. Prof. Dr. Berrin Yanıkoğlu ……

Assist. Prof. Dr. Hakan Erdoğan ……

DATE OF APPROVAL: ……

DESIGN, IMPLEMENTATION AND EVALUATION OF
A REAL-TIME P300-BASED
BRAIN-COMPUTER INTERFACE SYSTEM

Armağan Amcalar

Electronics Engineering, M.Sc. Thesis, 2010

Thesis Supervisor: Assist. Prof. Dr. Müjdat Çetin

Keywords: Brain-Computer Interface, P300 Speller

Abstract

In this thesis, we present a new end-to-end brain-computer interface system based on electroencephalography (EEG). Our system exploits the P300 signal in the brain, a positive deflection in event-related potentials, caused by rare events. P300 can be used for various tasks, perhaps the most well-known being a spelling device.

We have designed a flexible visual stimulus mechanism that can be adapted to user preferences. We have developed and implemented EEG signal processing, learning and classification algorithms. Our classifier is based on Bayes linear discriminant analysis, in which we have explored various choices and improvements. We have designed data collection experiments for offline and online decision-making. We have proposed modifications in the stimulus and decision-making procedure to increase online efficiency. We have evaluated the performance of our system on 8 healthy subjects on a spelling task and have observed that our system achieves higher average speed than state-of-the-art systems reported in the literature for a given classification accuracy.

P300 TABANLI GERÇEK ZAMANLI BİR BEYİN-BİLGİSAYAR ARAYÜZ SİSTEMİNİN TASARIM, UYGULAMA VE ANALİZİ

Armağan Amcalar

Elektronik Mühendisliği Yüksek Lisans Tezi, 2010

Tez Danışmanı: Yard. Doç. Dr. Müjdat Çetin

Anahtar Kelimeler: Beyin-Bilgisayar Arayüzü, P300 Heceleticisi

Özet

Bu tez çalışmasında, elektroensefalografi (EEG) tabanlı, yeni bir baştan sona beyin-bilgisayar arayüzü sistemi sunuyoruz. Sistemimiz, beyinde olaya-bağlı potansiyellerde oluşan P300 isimli artı yöndeki sinyali kullanmaktadır. P300 pek çok uygulama için kullanılabilmektedir. Bunların belki de en bilineni heceleme sistemleridir.

Kullanıcı tercihlerine göre şekillendirilebilen esnek bir görsel uyaran mekanizması tasarladık. EEG sinyal işleme, öğrenme ve sınıflandırma algoritmaları geliştirip uyguladık. Üzerinde çeşitli seçim ve iyileştirmeleri incelediğimiz sınıflandırıcımız Bayes doğrusal ayırtaç analizine dayanmaktadır. Bağlantısız ve çevrimiçi karar verme senaryoları için veri toplama deneyleri tasarladık. Çevrimiçi verimliliği artırmak için uyaran ve karar verme yordamlarında değişiklikler önerdik. Sistemimizin başarımını 8 sağlıklı denek ile bir heceletici üzerinde değerlendirdik ve sistemimizin literatürde yer alan sistemlerden belli bir sınıflandırma doğruluğunda daha üstün ortalama hıza ulaştığını gözlemledik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In today's world, controlling almost any object requires physical interaction with that object. For communication, we talk. For using a tool, we use our hands. When driving, we use our hands for controlling the wheel and our feet for controlling the pedals. Although researchers are trying hard to develop other means of control that won't need physical interaction, the opportunities are limited. Control of the environment, or communicating with other people are possible only for healthy human beings. People who lost their control over their limbs or other muscles have little chance for communication and control. For example, people with Amyotrophic Lateral Sclerosis (ALS), brainstem stroke or Multiple Sclerosis (MS) have damaged motor neural pathways. In such a case, voluntary control over the body is lost fully or partially, and the patient is locked into his/her body [41].

With technological advancements, however, new ways of communication open up for people. One of the most promising technologies is brain-computer interfacing. A brain-computer interface (BCI) is intended

to help disabled subjects gain control over their environment with the use of their brain activity. A computer maps the activity of the subject's brain to functions the subject is in need of like communication or physical control.

There are a number of techniques for recording activity from the brain, either invasively or non-invasively. Electroencephalography (EEG) is a noninvasive technique that records electrical brain activity via electrodes attached to the scalp of a subject. Along with EEG; magnetoencephalography (MEG), positron emission topography (PET), functional magnetic resonance imaging (fMRI) and optical imaging; functional near infrared spectroscopy (fNIRS) provide other ways to monitor brain activity [21].

In a BCI system that uses EEG, the computer records incoming signals from an EEG amplifier and by utilizing signal processing and classification techniques, analyzes and makes a decision of what to do with the data. Current studies allow patients to control robot arms or prostheses [25], communicate by selecting letters and words on a screen [8, 15, 27, 29], control a cursor [42], or control virtual reality applications [3, 4], etc.

1.1 Scope

This thesis focuses on dealing with problems in a popular BCI application that lets the subject type by choosing a letter among a matrix of

letters present on a screen. This application utilizes the P300 component of the event-related potentials that occur in the brain as a response to visual or auditory stimuli. This application is known as the P300 Speller and is first introduced by Farwell and Donchin in 1988 [15]. In this thesis, we present a new flexible stimulus mechanism that can be adapted to user preferences. We have designed data-collection experiments for offline and online decision-making. We have proposed modifications in the stimulus and decision-making procedure to increase online efficiency. Our classifier is based on Bayes linear discriminant analysis, in which we have explored various choices and improvements. We have evaluated the performance of our system on 12 healthy subjects and observed that our system achieves higher average speed than state-of-the-art systems reported in the literature for a given classification accuracy.

1.2 Motivation

The motivation for this thesis has two aspects. The first one is social; providing a means of communication for locked-in patients, who, otherwise have no chance of communicating with the outer world is invaluable. Any effort in improving current conditions for the handicapped is worthy.

The second aspect is, despite all the advances in technology and various research done in this field, there are many open questions that do

not have a satisfactory answer yet. The biggest problem is, a researcher in this field deals with the brain, the most complex and obscure part of the human body. A human being's actions are realized as a result of tiny electrical interactions between neurons in his brain, and the researcher wants to analyze these electrical variation. There are an estimate of 100 billion neurons in a human brain [40], and on the scalp, an electrode measures combined potentials of millions of neurons. The amplitude of the signal is very low, and the signal is prone to interference, especially from the mains electricity. The functioning of the brain is yet to be understood; more research is necessary to understand the interaction between neurons and behavior related to the use of BCIs [21]. Furthermore, although similarities exist, every brain is unique and subject variability is a big problem; performance of people vary on every technique. Also, the performance and responses of a specific person changes over time, vary from session to session, due to physical or mental condition of the subject; whether he is ill, happy, sad, tired, etc.

In the context of the P300 speller which was first proposed by Farwell and Donchin in [15], the issue of spelling rate is the main problem. Researchers are trying to speed up the system by tackling various aspects in the paradigm such as electrode selection, stimulus shape, timings, and presentation, data sampling, feature extraction, filtering, classification algorithm and other processing procedures.

Another problem is in making the system real-time. Can people use the system in real life, for communicating easily? If yes, what will be the performance of the system, and will it satisfy the needs, in other words, will it be a reliable channel for communication? How long do we have to train the system, or the subject?

The common problem among all EEG applications, skin preparation before attaching electrodes, the usability of the overall system, etc. is still a problem in the speller.

From this perspective, there is an obvious need for continuous improvement in terms of higher robustness, online adaptation, compensation for time-varying responses, transferring classifier or filter parameters from session to session [21].

1.3 Contributions

Our goal in this thesis was to explore ways to increase the performance, especially, the real-time (online) performance of the speller. Using a known classifier, Bayesian Linear Discriminant Analysis (BLDA), we focused on developing a real-time system. The difficulties faced in this problem are introduced and ways to overcome them are presented, along with their performance. Applying BLDA to a real-time system, we have explored various optimizations and choices in EEG signal processing, and

proposed a new, greedy, decision-making algorithm that gives feedback to the subject about his/her choice in real-time based on the probabilistic results of the Bayesian classifier.

Another main focus of this work was developing a new stimulus technique to boost online performance, where we put a huge effort. For this purpose, we have developed a unique, extensive, customizable stimulus software that allows the researcher to experiment on various schemes and options such as stimulus timing, matrix size, matrix contents, coloring, etc. The software is also planned to form a basis for stimulus software developed as a tool in research at SU BCI group.

Overall, with our technique, we report a performance gain over published work in both online and offline analysis.

1.4 Outline

Chapter 2 presents the necessary background information about BCI, P300 speller paradigm, stimulus software, feature extraction and classification techniques and proposes mathematical preliminaries for the classifier.

Chapter 3 covers in detail technical features of the stimulus software we developed and used in our analysis. This chapter can also be used as a reference manual for the software.

Chapter 4 is about the offline experiments we have conducted with our subjects. Offline analysis method for the P300 speller, performance metrics and results of our experiments are given.

Chapter 5 discusses the problems that arise in online experiments, and our classification methods and decision making algorithms. The overall performance of our subjects is also reported.

Chapter 6 summarizes our work and presents our results in a compilation. A concluding discussion, and propositions and extensions about future work in the scope of this thesis are presented.

CHAPTER 2

BACKGROUND

This chapter intends to help the reader understand basic concepts about brain-computer interfaces, EEG signal processing and the P300 speller. A survey about published work, methods and results are also presented.

2.1 Introduction

Brain-computer interfaces help restore function to people with motor impairments by providing the brain with a new, non-muscular communication and control channel [41]. These systems monitor brain activity. As mentioned in Chapter 1, there are a number of alternatives for measuring brain activity that are used in the clinic to diagnose and track neurological disorders. These methods can be listed as electroencephalography (EEG), magnetoencephalography (MEG), positron emission tomography (PET), functional magnetic resonance imaging (fMRI)

and functional near infrared spectroscopy (fNIRS). For BCI approach, a method that is both fast, reliable and easy to acquire and use is required, so that patients can effectively use it. However, MEG, PET, fMRI and fNIRS are technically demanding, expensive and hard to utilize outside a laboratory. In contrary, EEG is relatively cheap, and offers different paradigms as control mechanisms.

As a communication and control channel, a BCI makes it possible for the handicapped to interact with the outer world, either by spelling letters and words, or selecting among a menu of medication needs; or to control the conditions of the room the person is in such as lighting or air conditioning, or prostheses for various tasks such as a motorized wheel chair, etc.

2.2 EEG

2.2.1 Basics

Electroencephalogram (EEG) is a technique for recording electrical activity of the human brain. Since the first introduction in humans by Berger in 1929 [5], EEG has been mainly used for clinical diagnosis of neurological disorders. EEG utilizes electrodes for recording brain activity. Recording via electrodes attached to the scalp is the most common and easy to apply noninvasive technique for EEG. Other invasive methods include

epidural, subdural or intracortical electrodes [41]. Invasive methods have an excellent resolution on the electrical activity of the brain, but are harder to implement and experiment with since they require surgical operation. This practical limitation drives most of the research in this area to be done with scalp recording, which is accomplished with electrodes attached to the subject's scalp. The main problem of this noninvasive recording is the amplitude of the recorded signals, which usually lies in 5-20μV range. Furthermore, a signal read from an electrode is the combination of activity of millions of neurons, so in general, is not very informative about the nature of the brain.

2.2.2 Electrodes

Electrodes, simply, little flat pads of Ag/AgCl, are attached to the scalp with the help of an elastic cap, known as the electrode cap, an example of whom is shown in Figure 2.1. To decrease skin resistance or voltage offset and to have a stable, stationary conductive medium for proper measurements, usually a conductive gel is applied to the skin after abrasive skin preparation. Due to the passive nature of the electrodes; reasons such as electromagnetic interference, noise and signal degradation, need for skin preparation, etc., are problems for practical usage of these electrodes outside the laboratory.

Figure 2.1 64-channel electrode cap featuring international 10-20 system for electrode distribution [7]

Instead, active electrodes were developed, in which the electrodes are combined with a small, very high input impedance circuitry that usually does amplification on spot (Figure 2.2). These electrodes offer higher resistance to interference, due to the fact that they buffer the received signal before it travels a long way to the actual EEG amplifier. Although 'dry' versions of these electrodes are proposed where conductive gel is not needed, the stability of their performance is of question. The electrodes are placed on the head of the subject according to an international system called 10-20 system, proposed by American EEG society[26]. This system proposes that the electrodes are placed in a 10% or 20% distance from each other with

respect to the total distance between the nasion and inion of the subject. Figure 2.3 shows the layout for a 64 channel EEG system. This is the system we follow in our recordings.

Figure 2.2 Active electrodes

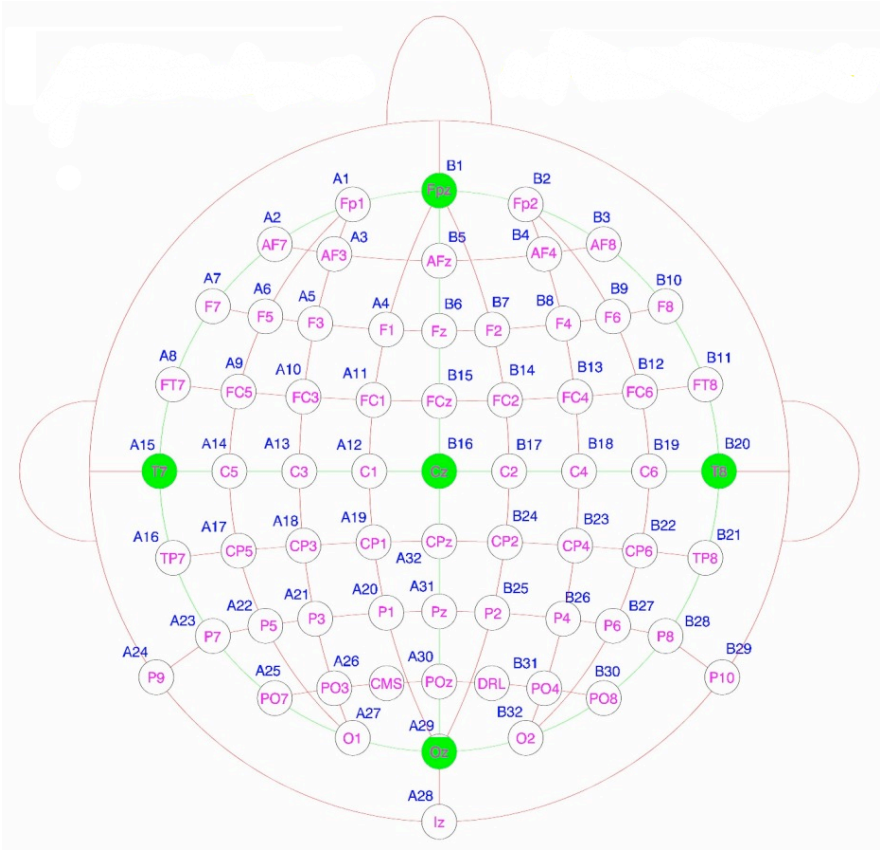


Figure 2.3 Electrode placement layout according to 10-20 electrode system [6]

Note that these active electrodes are pin-type, instead of the general flat electrodes, so that the electrode can reach to the skin when attached to the cap.

2.3 BCI Systems Details

A model of a typical BCI system is given in Figure 2.4. Basically, a BCI system begins with electrodes. Either active or passive, these electrodes transmit the electrical activity on the scalp of the subject to a high-sensitivity, low-noise amplifier, namely the EEG amplifier.

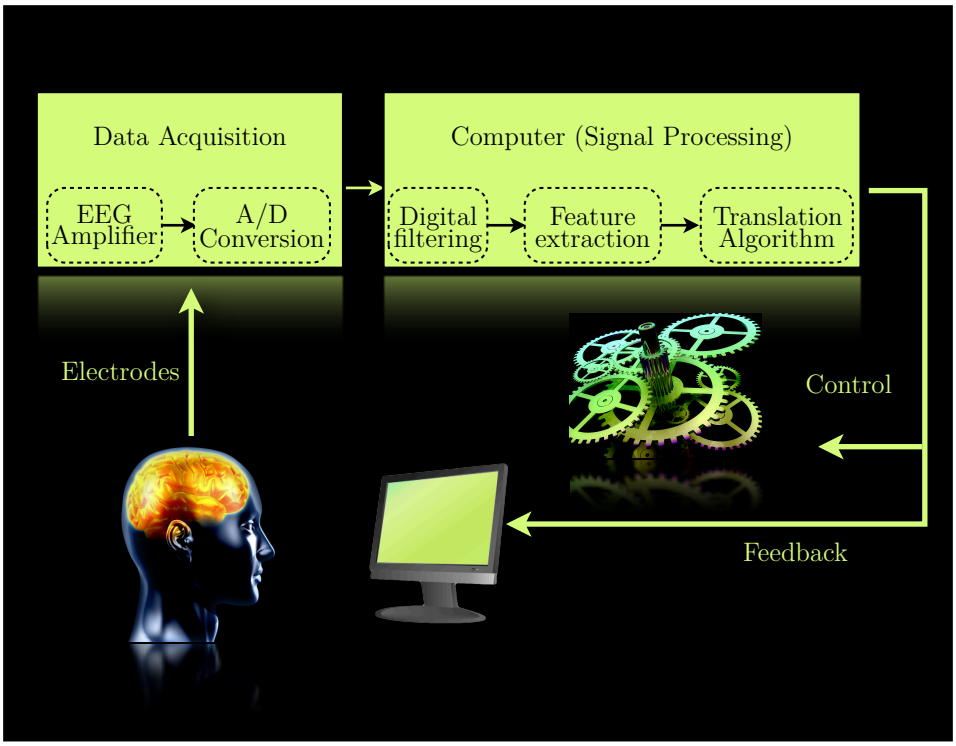


Figure 2.4 A general BCI system model

EEG amplifier applies basic analog filtering such as a high-order notch filter to remove mains interference, a high-order low-pass filter to remove

DC components, a high-order high-pass filter to remove irrelevant frequency components. Some EEG amplifiers do not feature a power input for mains, they just work on batteries. This helps in blocking interference. EEG amplifier also amplifies the signals a few thousand times, until the signals are at an appropriate level for the analog to digital converter to sense. Generally, to prevent crosstalk between analog and digital data, ADC is separated from the analog amplifiers and has a box of its own.

Output of the ADC is fed to a computing device. While this device might be an embedded platform in the case of a BCI product, generally, it is a common PC that runs the researcher's analysis software.

A typical raw EEG signal recorded for a duration of 1 s from channels at location Fz, Cz, Pz and Oz is depicted in Figure 2.5. Note that the signals include very high frequency components, usually out of the scope of EEG analysis. The amplitude of these components are generally large enough to block the frequency spectrum of interest. Therefore, the researcher further filters the incoming data to suit his/her purposes.

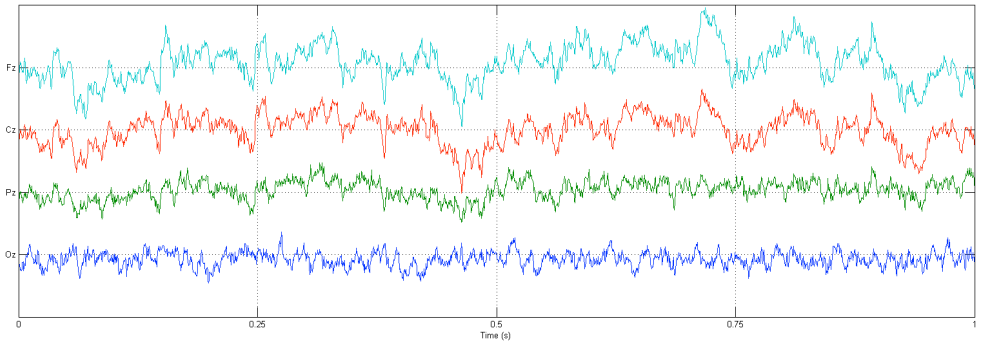


Figure 2.5 A typical raw EEG signal recorded from locations Fz, Cz, Pz, Oz

Different BCI approaches have different interests in the frequency spectrum. Therefore a digital filtering process follows next. In the scope of this thesis, signals between 1-12Hz are investigated. Figure 2.6 shows the same epoch filtered with a 6^{th} order Butterworth bandpass filter.

After filtering the data, the researcher looks for some specific features in it. The process that finds these features is called feature extraction. Features might be peaks, actual or special waveforms or deflections at specific times, spectral density, etc. In the scope of this thesis, the features are almost an imitation of the actual waveform, in other words, the amplitude of the signals for that period. Figure 2.7 depicts the same epoch after feature extraction. Figures 2.6-7 were shrunk for a better view of the features.

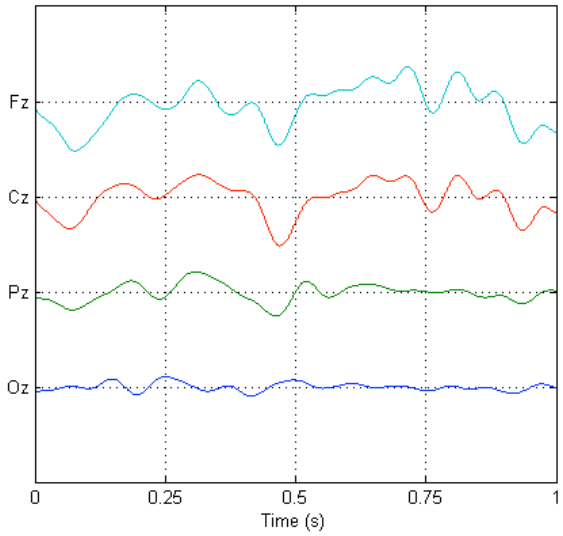


Figure 2.6 Bandpass filtered epoch

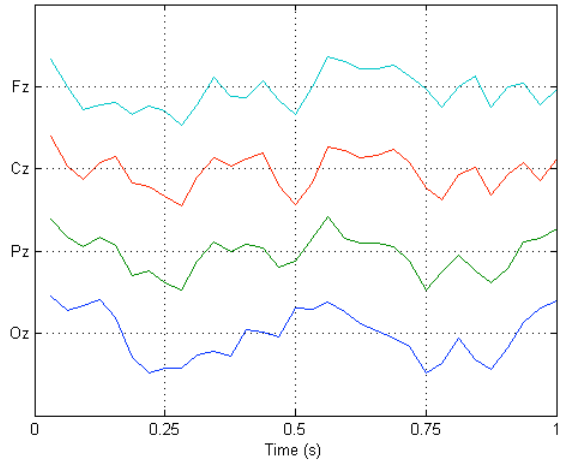


Figure 2.7 Feature extracted epoch

After feature extraction, a feature vector is formed. This feature vector is then run through a classifier, and that classifier decides what to do with the data. For example, in our study, we want to know if the feature vector includes signs of a P300 wave. If it does, we acknowledge that the subject is gazing on a letter he/she wants to type. If required conditions are satisfied, we might show the letter to the user as a feedback. This algorithm that decides what to do with the data (going on sampling, or finding a target) is called the translation algorithm, and might include one or more classifiers. In another BCI application, rotation of a robotic arm might be controlled by the spectral power of alpha waves. In other words, spectral power might be translated into a parameter for rotating a robotic arm. Either by feedback, or controlling some outer device, the result of the analysis realizes a function the subject is in need of.

2.4 Event Related Potentials, P300 component

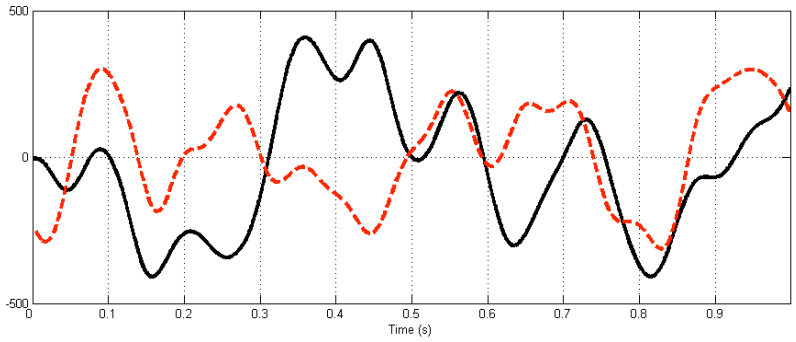
Event related potentials (ERP) occur in the brain as a response to an external event. They can be measured before, during or after a sensory, motor or psychological event [11, 21] and usually have a fixed time delay after (or before) the event, named stimulus. In 1964, Walter et al. discovered that when a subject was required to press a button after detecting a target in a visual stimulus, they elicited a large negative voltage at frontal

electrodes that happen just before the subject presses the button [38]. This voltage, ERP component called Contingent Negative Variation (CNV) indicated the subject's mental preparation to press the button. In 1965 Sutton et al. discovered the P3 component, a large positive deflection in brain signals that occur around 300 ms after the stimulus in an experiment where subjects were presented with either auditory or visual stimulus in a random fashion where the subjects could not predict what the next stimulus would be [35]. In contrary, when the modality of the stimulus was perfectly predictable, the P3 component was much smaller in amplitude.

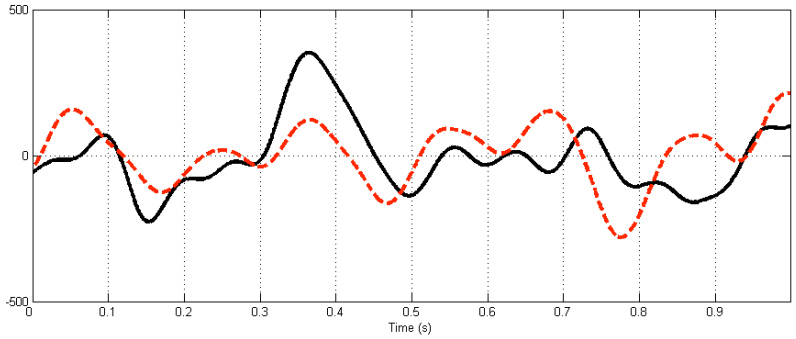
This idea brought about another paradigm known as the 'oddball paradigm', where the subject is stimulated with two categories of events - relevant and irrelevant. The relevant events occur rarely with respect to irrelevant events, and due to the complete random order of events, elicit a large P300 response in ERPs. In 1988, Farwell and Donchin used this paradigm to develop a communication system where subjects were able to type letters on a computer screen only by thought - with P300 signals [15]. Farwell and Donchin present a 6x6 matrix of letters and numbers to the subject. The rows and columns of the matrix flash in a block-randomized fashion, and the user is required to keep a mental count of the number of occurrences of a target, a letter he/she wants to type. Here, the row and column that contain the target letter are the relevant events, where in a block of 12 flashes, there are two of such events. The other events, rows and

columns that don't interest the subject are irrelevant events, and there are ten of such events in a 12-flash block.

Figure 2.8 shows typical P300 responses of single trials, and averaged trials recorded at electrode site Pz.



(a)



(b)

Figure 2.8 (a) Single trial epochs; solid line is the response to a target stimuli, dashed line is the response to irrelevant stimuli. (b) Average of 10 trials; solid line is the averaged response to 10 target stimuli, dashed line is the averaged response to 10 irrelevant stimuli

This phenomenal research pointed that in the end, P300 wave can be used as an efficient actor in a new communication channel, a BCI. The proposed method opened a wide area of possibilities for further research and brought up countless new questions which, after 22 years, are yet to be answered.

One of the key elements in this research is how the stimulus is delivered to the subject. The next section underlies basic principles for common P300 stimulus software and includes properties of a well-known and widely used software package called BCI 2000.

2.5 P300 Stimulus Software

A stimulus software has many purposes. Undoubtedly, the main purpose is to deliver the subject the required visuals, or directions, to evoke the necessary potentials. Apart from that, it is the duty of the stimulus software to note the exact time frames when the stimuli occurred, since the analysis in BCI depends solely on the actions of the subject. In other words, an action can only be analyzed when one can extract its response epoch from the recordings. On BCI systems, this is achieved by triggering. The stimulus software interfaces the recording device and sends a trigger signal whenever an action takes place. It is the duty of the device to save this

information on a separate data channel. Aforementioned epochs are extracted from the recordings with this trigger information.

Since the needs of every research and researcher are different, research groups tend to develop their own stimulus software, trimmed to their special needs. As a research group may want to obtain different results based on small variations in the stimuli, the software might be customizable, or even extendible through plug-ins, etc.

2.5.1 Common P300 Speller Software Properties

A P300 speller software features visuals on screen that flash for predefined durations at predefined intervals. These flash events are called trials. Commonly, a matrix of visuals are presented by rows and columns. In other words, in any trial, a row or a column flashes. There are always two targets in a run, namely a row and a column that intersect, where, at the intersection is the target visual for the subject (*see section 4.3 for speller terminology*). The rows and columns are intensified in a block-randomized fashion, where some limitations might be applied (see sections 3.2.3 for a possible case of limitation). The rule in block-randomization is that, after a trial group of m+n intensifications, every row and every column is intensified only once. Therefore, before moving to the next trial group, we make sure that every element was intensified for an equal amount.

2.5.2 BCI 2000 P300 Speller

BCI 2000 is a complete set of tools used by EEG research groups all over the world. Featuring a module-based system, BCI 2000 has the capability of data acquisition from several hardware, two stage (feature extraction and feature translation) signal processing phase, application interface where the subject decides an action with the help of translated control signals, and an operator interface to set various parameters and monitor other software and/or experiment related information [30].

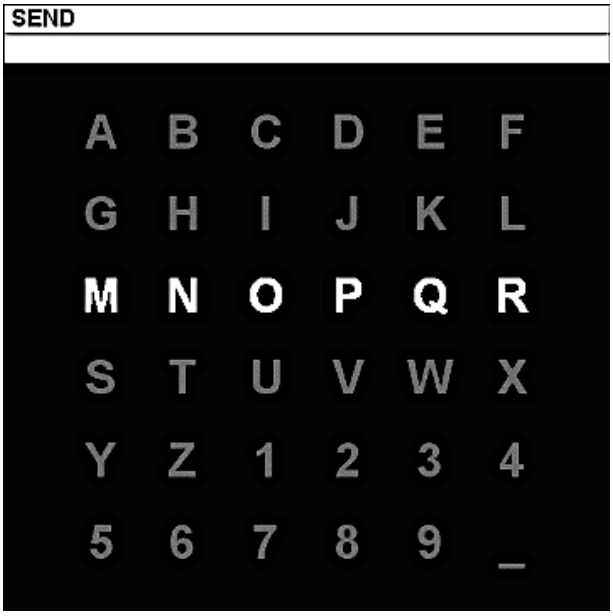


Figure 2.9 Typical screenshot of BCI 2000 P300 speller application. Top box (including word SEND) is for specifying the target letters in copy mode. The empty box below it is for feedback.

As a part of the User Application Module, BCI 2000 features a P300 based speller application. Detailed in [1], this application has a wide variety of options to suit most P300 speller needs. Defining custom matrix dimensions, nested matrices, ability to show bitmaps in matrix cells instead of letters, or choosing a .wav file for auditory stimuli and feedback of classification operations are some of the functionalities that make BCI 2000 so popular among researchers.

BCI 2000 speller can both be used in offline and online mode. In the copy (offline) mode, the operator is free to enter any text to be spelled, and the software pursues that text during the session. In the free (online) mode, there are no predefined text and the subject is free to choose any cell to concentrate on. In this mode, the software gives feedback about the detected choice of the subject.

Overall, with its extensive options, BCI 2000 provides a quick solution for any research group working with P300 signals.

2.6 Survey of techniques

This chapter intends to inform the reader about classification and feature extraction approaches used in P300 BCI context.

2.6.1 Stepwise Discriminant Analysis (SWDA)

Stepwise linear discriminant analysis is a technique for selecting suitable (predictor) variables (usually amplitude values) among the data to be included in a multiple regression model [20]. Initially starting with no terms in the model, forward stepwise regression is applied and the most statistically significant predictor variable having a p-value smaller than 0.1 is added to the model. After each new entry to the model, a backward stepwise regression is performed and the least significant variables having p-values greater than 0.15 are removed. This process is repeated until the model includes a predetermined number of terms, or until all the terms that satisfy the entry/removal criteria are processed. This technique is applied in [12, 15, 31] and results are reported.

2.6.2 Support Vector Machines (SVMs)

SVM is a kernel-based classification method. Given a two class classification problem, SVM tries to find the optimum separating hyperplane (OSH) (or vector in this case) that has the maximum distance between itself and the nearest samples in each class [14]. These samples constitute the support vector; hence the name. This approach is good only for linearly separable samples, but real life examples such as EEG data are quite inseparable from each other because of the background activity of the

brain, and so nonlinear solutions have to be applied. Therefore, kernel functions, which map vectors into a higher dimensional space, are utilized to introduce nonlinearity to the optimum separating hyperplane. Polynomial kernels, Gaussian Radial Basis Functions and Exponential Radial Basis Functions are among the most popular kernel functions used in EEG. [36, 39] show example uses and report performance values using SVMs.

2.6.3 Fisher's Linear Discriminant Analysis (FLDA)

FLDA aims to compute a discriminant vector that separates two or more classes as well as possible. In this study, we only have two classes, one for the case that epoch under investigation includes a P300 wave, and one for the case that epoch under investigation does not include a P300 wave. FLDA searches for discriminant vectors that result in a large distance between the projected means and small variance around the projected means (small within-class variance). The discriminant vector $\boldsymbol{w}$ is a function of these data, and the output of the analysis, given an input vector $\hat{\boldsymbol{x}}$, is simply $\boldsymbol{w}^{\mathsf{T}}\hat{\boldsymbol{x}}$ [17]. The output values generated by this analysis may be handled in two ways; for single trial EEG analysis, the maximum of the output values contains the answer, or for better accuracy, the output might be summed over multiple trials, and then the maximum can be chosen.

2.6.4 Bayesian Linear Discriminant Analysis (BLDA)

BLDA can be seen as an extension to Fisher's Linear Discriminant Analysis (FLDA). In BLDA, regularization is used to prevent over-fitting problem. A possible solution for finding regularization parameters is cross-validation, which is a time-consuming and stationary process. Instead, with a Bayesian analysis, the degree of regularization can be estimated automatically and quickly from training data [17].

In the next section mathematical preliminaries of BLDA is presented.

2.6.5 Principal Component Analysis (PCA)

First described in 1901 by Pearson, PCA is a popular technique used in dimensionality reduction. PCA is a linear transformation that maps a feature space of correlated variables to a higher space of uncorrelated variables (hence the name principal components). In other words, PCA retains components of the dataset that contribute most to its variance, while ignoring the rest, where by definition, those components contain the 'most important' aspects of the data [2]. PCA works best if individual components have Gaussian distributions. PCA is a widely used technique both in dimensionality reduction and in classification. An example work that uses PCA is given in [39].

2.6.6 Independent Component Analysis (ICA)

Independent Component Analysis is another technique used in decomposing a signal which is a combination of many other 'independent' signals (also known as the blind signal separation). Independence is a much stronger property than uncorrelatedness, so PCA is unable to separate these independent components. ICA is successfully applied in EEG signals. An example where ICA is used, along with its results can be seen in [23, 33].

2.7 BLDA

This section exactly follows Appendix B of [17] where a summary of BLDA is given. For a more detailed explanation see [16]. Algorithms that are closely related to the method presented below are the Bayesian least-squares support vector machine [37] and the algorithm for Bayesian non-linear discriminant analysis described by [10]. BLDA is also closely related to the so-called evidence framework for which detailed accounts are given in [22] and [9].

The fact that FLDA is a special case of least squares regression if regression targets are set to $\frac{N}{N_1}$ for examples from class 1 and to $-\frac{N}{N_2}$ for examples from class -1 (where N is the total number of training examples,

N_1 is the number of examples from class 1 and N_2 is the number of examples from class -1). [9] gives a proof for the equivalence between least squares regression and FLDA. Given the connection between regression and FLDA, BLDA performs regression in a Bayesian framework and sets target values as mentioned above.

The assumption in Bayesian regression is that targets t and feature vectors $\boldsymbol{x}$ are linearly related with additive white Gaussian noise n.

$$t = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + n \tag{1}$$

Given this assumption, the likelihood function for the weights $\boldsymbol{w}$ used in regression is:

$$p(\mathbf{D} \mid \beta, \boldsymbol{w}) = \left(\frac{\beta}{2\pi}\right)^{\frac{N}{2}} \exp\left(-\frac{\beta}{2} \mid\mid \mathbf{X}^{\mathsf{T}}\boldsymbol{w} - \boldsymbol{t} \mid\mid^2\right) \tag{2}$$

Here $\boldsymbol{t}$ denotes a vector containing the regression targets, $\mathbf{X}$ denotes the matrix that is obtained from the horizontal stacking of training feature vectors, $\mathbf{D}$ denotes the pair $\{\mathbf{X}, \boldsymbol{t}\}$, β denotes the inverse variance of the noise and N denotes the number of examples in the training set. It is assumed that the feature vectors contain one feature which always equals one; the bias term which is commonly used in regression can thus be omitted.

To perform inference in a Bayesian setting, one has to specify a prior distribution for the latent variables, i.e. for the weight vector $\boldsymbol{w}$. The expression for the prior distribution is:

$$p(\boldsymbol{w} \mid \alpha) = \left(\frac{\alpha}{2\pi}\right)^{\frac{D}{2}} \left(\frac{\varepsilon}{2\pi}\right)^{\frac{1}{2}} \exp(-\frac{1}{2}\boldsymbol{w}^{\mathsf{T}}\mathbf{I}'(\alpha)\boldsymbol{w}) \ , \tag{3}$$

where $\mathbf{I}'(\alpha)$ is a square, $D+1$ dimensional, diagonal matrix

$$\mathbf{I}'(\alpha) = \begin{bmatrix} \alpha & 0 & \cdots & 0 \\ 0 & \alpha & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \varepsilon \end{bmatrix}, \tag{4}$$

and D is the number of features. The prior for the weights thus is an isotropic, zero-mean Gaussian distribution. The effect of using a zero-mean Gaussian prior for the weights is similar to the effect of regularization term used in ridge regression and regularized FLDA. The estimates for $\boldsymbol{w}$ are shrunk towards the origin and the danger of over-fitting is reduced. The prior for the bias (the last entry in $\boldsymbol{w}$) is a zero-mean univariate Gaussian. Setting ε to a very small value, the prior for the bias is practically flat. This expresses the fact that a priori there are no assumptions made about the value of the bias parameter.

Given likelihood and prior, the posterior distribution can be computed using Bayes rule.

$$p(\boldsymbol{w} \mid \beta, \alpha, \mathbf{D}) = \frac{p(\mathbf{D} \mid \beta, \boldsymbol{w}) p(\boldsymbol{w} \mid \alpha)}{\int p(\mathbf{D} \mid \beta, \boldsymbol{w}) p(\boldsymbol{w} \mid \alpha)\, d\boldsymbol{w}}\,, \tag{5}$$

Since both prior and likelihood are Gaussian, the posterior is also Gaussian and its parameters can be derived from likelihood and prior by completing the square. The mean $\boldsymbol{m}$ and covariance $\mathbf{C}$ of the posterior satisfy the following equations.

$$\boldsymbol{m} = \beta(\beta \mathbf{X}\mathbf{X}^{\mathsf{T}} + \mathbf{I}'(\alpha))^{-1}\mathbf{X}\boldsymbol{t} \tag{6}$$

$$\mathbf{C} = (\beta \mathbf{X}\mathbf{X}^{\mathsf{T}} + \mathbf{I}'(\alpha))^{-1} \tag{7}$$

By multiplying the likelihood function (equation 2) for a new input vector $\hat{\boldsymbol{x}}$ with the posterior distribution (equation 5) followed by integration over $\boldsymbol{w}$, we obtain the predictive distribution, i.e. the probability distribution over regression targets conditioned on an input vector:

$$p(\hat{t} \mid \beta, \alpha, \hat{\boldsymbol{x}}, \mathbf{D}) = \int p(\hat{t} \mid \beta, \hat{\boldsymbol{x}}, \boldsymbol{w}) p(\boldsymbol{w} \mid \beta, \alpha, \mathbf{D})\, d\boldsymbol{w} \tag{8}$$

The predictive distribution is again Gaussian and can be characterized by its mean μ and its *variance* σ^2.

$$\mu = \boldsymbol{m}^{\mathsf{T}} \hat{\boldsymbol{x}} \tag{9}$$

$$\sigma^2 = \frac{1}{\beta} + \hat{\boldsymbol{x}}^{\mathsf{T}} \mathbf{C} \hat{\boldsymbol{x}} \tag{10}$$

In this study, only the mean value of the predictive distribution was used for taking decisions. More precisely, mean values were summed over trials and the letter corresponding to the maximum of the summed mean values was selected as the answer of classification.

In a more general setting, class probabilities could be obtained by computing the probability of the target values used during training. Using the predictive distribution from equation 8 and omitting the conditioning on $\boldsymbol{\beta}, \boldsymbol{\alpha}, \mathbf{D}$ we could use:

$$p(\hat{y} = 1 \mid \hat{\boldsymbol{x}}) = \frac{p(\hat{t} = \frac{N_1}{N} \mid \hat{\boldsymbol{x}})}{p(\hat{t} = \frac{N_1}{N} \mid \hat{\boldsymbol{x}}) + p(\hat{t} = \frac{-N_2}{N} \mid \hat{\boldsymbol{x}})} \tag{11}$$

Both the posterior distribution and the predictive distribution depend on the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Although in this summary these parameters are assumed to be known, in real-world situations the hyperparameters are usually unknown. One possibility to solve this problem would be, as previously stated, to use cross-validation to determine the hyperparameters that yield the best prediction performance. However, Bayesian regression framework offers a more elegant and less time-consuming solution for the problem of choosing the hyperparameters. The idea is to write down the likelihood function for the hyperparameters and

the maximize the likelihood with respect to the hyperparameters. The maximum likelihood solution for the hyperparameters can be found with a simple iterative algorithm which is described in [22]. For a detailed explanation of BLDA, the user is encouraged to read [16].

2.8 State-of-the-art performance

Previous works have firmly shown that a P300 speller is an efficient communication channel, especially for disabled people who have no other means of communication. In this section, we will briefly discuss results reported by other researchers. To begin with, in [15], the original paper of Farwell and Donchin that introduced this speller, the typing rate is reported as 2.3 letters per minute with 95% accuracy. Although the bit rate (explained in Chapter 4) is reported as 12bits/min in this work, we calculated it as 10.67 bits per minute. Later, Donchin increased this rate up to 4.3 letters/min with 95% accuracy in [12]. The bit rate for this work was calculated as 19.83 bits/min. Meinicke reported 5.5 letters/min with above 90% accuracy in [24], with a roughly calculated bit rate of 24 bits/min. In [18], Kaper reported an average of 47.26 bits/min. Serby reported an average of 5.45 letters/min with 92.1% accuracy and 23.77 bits/min in [33].

On top of these results, we report that with 6 able bodied subjects, we have reached an average online performance of 12.48 letters/min with

89.75% accuracy and 11.14 letters/min with error compensation, which is still prone to errors but makes sure that the subject types all the letters he/she wants. Therefore, the time that wrong results take for classification is included and counted as lost time. For offline analysis, we report a bit rate of 56 bits/min. As far as we know, the speed of our system is unmatched among previously published work. The next chapters will give a lot more detail about our methodology and results.

CHAPTER 3

STIMULUS SOFTWARE

As previously stated, a P300 BCI implementation requires many components, one of which is a stimulus software. Stimulus software is an essential part of BCI systems, where the analysis depends on processing signals that evoke after a subject is required to take an action. In this chapter, we describe in detail the SU-BCI P300 stimulus software we have developed.

3.1 SU-BCI P300 Stimulus Software

3.1.1 Basics

The stimulus software we developed for our research group is essentially a matrix-based system, first introduced by Farwell and Donchin in [15]. The software has many customization options that allows the user to

derive numerous analyses and cross-analyses within the context of a P300 speller.

Ease of use, a simple and clean user-interface and customizability were the design goals for the software. Since SU-BCI group has plans for further studies in P300 speller context, the software had to be able to satisfy diverse needs. With this clear goal in mind, the software architecture is built so that the broad needs of different P300 experiments can be satisfied within a single software, by changing settings and parameters.

The software is developed in C#, an object-oriented language based on C++. While having advanced OOP features and capabilities that include and surpass those of C++, C# has a visual form management as easy as Visual Basic. The main motivations for developing the software in C# are the ease and quickness of development, ability to further develop extensions to allow more customization, and the ability to interface with our BioSemi hardware for triggering purposes.

3.1.2 Features and Preferences

The software is used on every experiment throughout the work in this thesis. The major advantages of the software are the ability to create a visual matrix of any shape in principle, the possibility of choosing any intervals for flashes, adjustable countdown timer for the target letter, and

the ability to adjust how many trial groups (see section 4.3 for an explanation of a trial group) will be used for one letter. All of these settings are accessible through the options panel in the top section of the interface.

In the following sections, detailed explanation of each option in the options panel and feature will be provided.

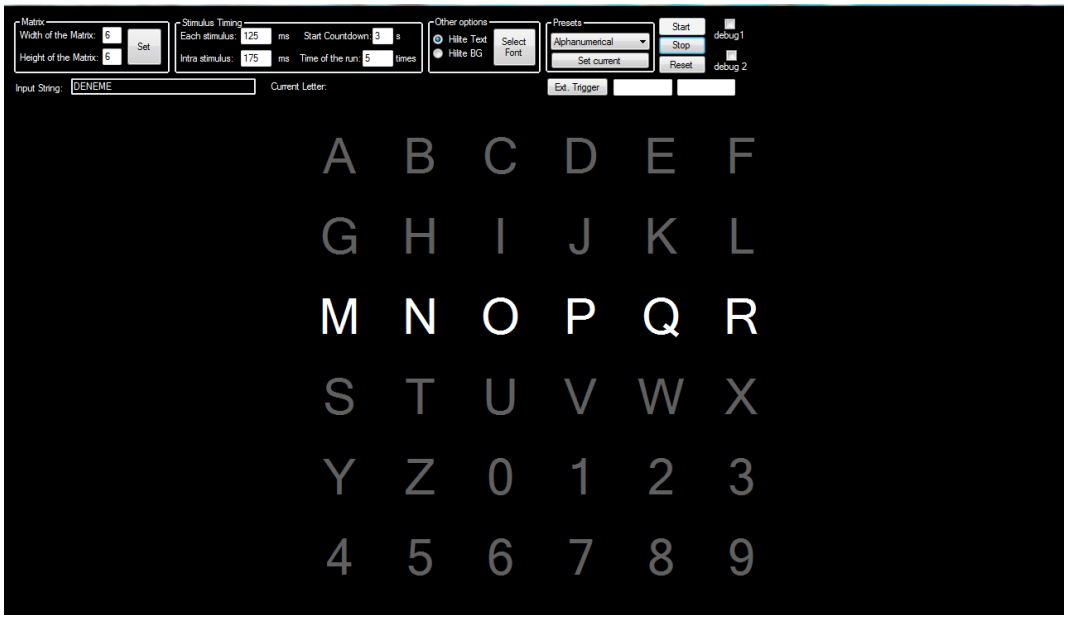


Figure 3.1 Software screenshot, 6x6 matrix in action, 3rd row intensified

3.1.2.1 Matrix dimensions

The most important detail in a P300 speller stimulus is the matrix dimension. Matrix dimensions have direct effect on the performance of the analysis, due to the oddball paradigm fundamentals. Research about the

effects of different matrix sizes is essential and ongoing, as exampled in [13] & [32].

To allow future studies of changes of matrix dimensions, the software lets the conductor to define the width and height parameters of the matrix as the first element in the options panel. Theoretically, any positive integer is allowed for each dimension, as the software automatically fits (either by enlarging or reducing) the matrix in the window, although a row or column count of more than 10, might be inconvenient for the subject, as the number of visuals increases and therefore each visual becomes harder to detect.

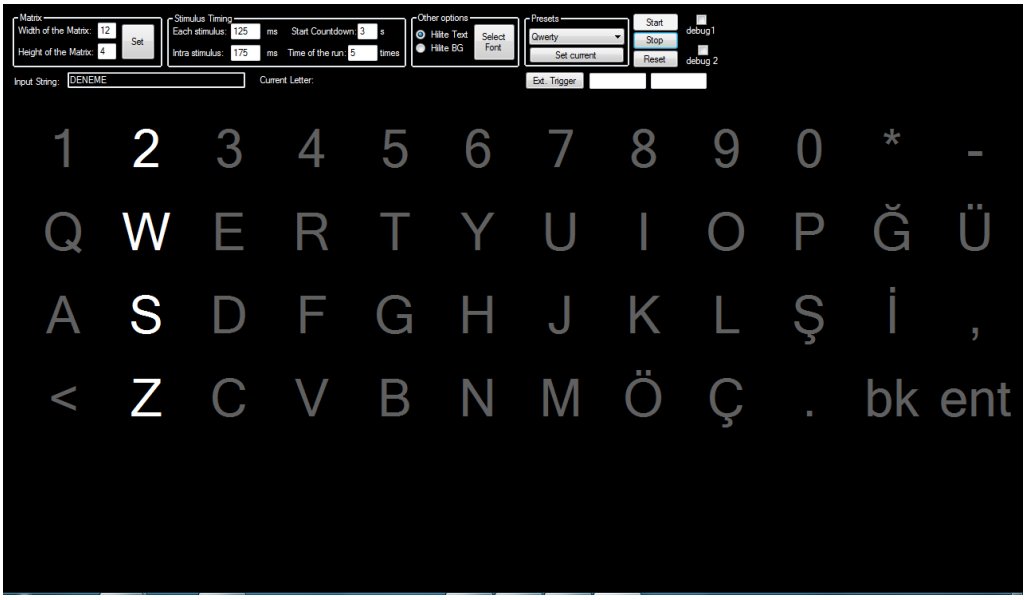


Figure 3.2 A 4x12 sized matrix, 2nd column intensified

3.2.2.2 Matrix elements (visuals)

The conductor is free to enter any letter or text in each cell of the matrix, in any order. Furthermore, the software allows the conductor to use symbols in a matrix, by letting the conductor choose any font-face for the matrix, including *dingbats* (symbol-based fonts) such as Zapf Dingbats, Wingdings and Webdings. In addition, since the software allows to use any font installed in the system, one can create a font solely to define a stimulus, making the capabilities of the software creating matrices virtually infinite.

Since the conductor enters the contents of each cell by hand, he is free to choose not to enter any visual in a cell. With this approach, one can design an asymmetrical matrix, or even a round matrix of visuals, as shown in Figure 3.3.

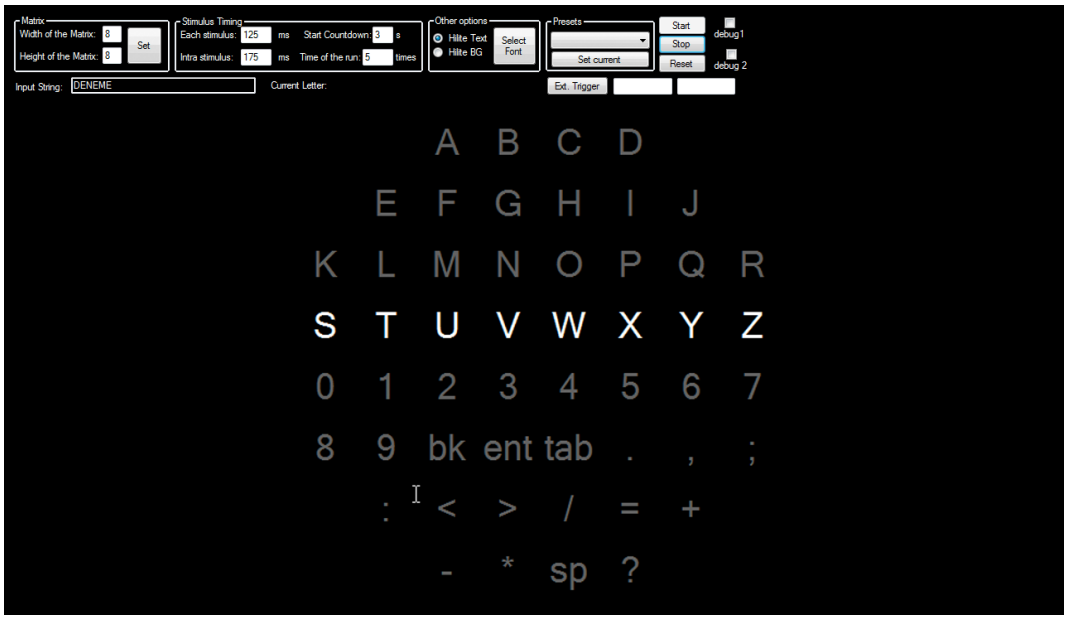


Figure 3.3 A round matrix. Although the matrix is 8x8, some elements at each corner are left blank to get a "round" shape.

3.2.2.3 Presets

By default, the conductor is required to define a new matrix each time the software is run. This is a serious limitation as it might take valuable time. Therefore we have given the conductor the option of presets, hard-coded predefined matrices.

A preset is a set of hard-coded variables in the source code of the software. A preset consists of the width and the height of the matrix, as well as the visuals inside. These hard-coded options are selectable in the Presets frame in the options panel. Although we have conducted all our experiments

with the first preset, the 6x6 matrix, another matrix, the Turkish QWERTY keyboard is also provided by default (see Fig. 4.2).

The easy way of preparing for the session is of course, preparing a preset and choosing it at the beginning of each session. With this opportunity, the conductor has the matrix prepared with one click.

3.2.2.4 Stimulus timing

One of the key elements affecting the performance of a P300 speller method is the stimulus timing, as discussed in [32] and [39].

To allow this kind of analyses on stimulus timing, the software has the Stimulus Timing frame in the options panel. Stimulus timing has two elements, span of each stimulus (denoted by *Each stimulus*), and the period between two stimuli (denoted by *Intra-stimulus*). Span of each stimulus specifies for how long a visual will be intensified, and intra-stimulus period decides the temporal gap between two intensifications.

Other options in this frame allow further customization. *Start Countdown* specifies for how long the runs will be delayed at the beginning of each session. As another important parameter, this period allows the subject a break, for long runs might falter the concentration of the subject. Moreover, the conductor might need some time to leave the room after starting the session, or might need some time for other arrangements.

The last, and the most important parameter in this group is for specifying how many trial groups there will be in a run, as this number directly affects classification performance. This effect has clearly been observed and reported in [17], [31] and [36]. In principle, the conductor is free to choose any number of trial groups, but the classification performance is generally satisfactory after 10 trial groups. The conductor should keep in mind the fact that longer runs negatively affect performance, as evidenced by the average P300 wave amplitude as shown in [19], [28] and [31].

3.2.2.5 Other options

Other options include font preference, and a minor style option called highlighting. Highlighting defines the style of the visual during intensification. There are two options; Highlight Background and Highlight Visual, and the conductor is able to choose one of them for the experiment. At startup or during intra-stimulus period, cell backgrounds are black and all the visuals are dark gray. During intensification, a row or a column becomes highlighted, i.e. turns white. These two options specify which part of the visual will be white, the cell background, or the visual itself. Examples are shown below.

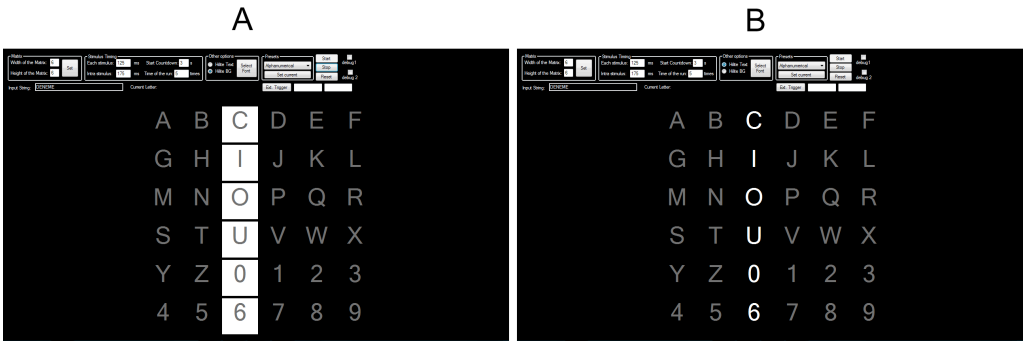


Fig 3.4 Different highlighting modes. (A) Background highlighting (B) Highlighting of the visuals

As described before, with this software, the conductor is able to design a matrix that uses figures and shapes. This is achieved by different font types called Dingbats. The conductor is able to define the font-type with the Select Font button. Once selected, all the entered text in the matrix will be converted into that selected font.

Figure 3.5 2x2 matrix with shapes

3.2.2.6 Control commands

These options control the basic functionality of the stimulus session. To begin a session, the conductor clicks start. If all the parameters are set to accepted values, the session begins. In case of a parameter value error, the software just quits.

If no user intervention occurs, a session ends at the projected time, returning the software to the first state where the conductor is again allowed to make changes.

If the conductor needs to cancel the ongoing session, he can do so by clicking stop. Afterwards, to begin a new session, the conductor has to reset the session progress by clicking Reset.

3.2.2.7 Extra features

The software has some more auxiliary features implemented for debugging purposes. Two debug modes allow monitoring the operation of the software. Mode one records every event in the software. It notes down the exact beginning time of each run, target letter for each run and the intensifications by row and column numbers. This mode is useful for checking the recorded data and making sure it was recorded as intended, at the intended time.

The second mode clears the matrix, resetting all the values of the cells to 0. Whenever a row or a column is intensified, all the elements belonging to that row or column get an increase in value. This mode is useful for checking that the algorithm is correctly distributing intensifications in groups of trials. It is also useful for understanding the randomized working principles of the algorithm, if one chooses to implement a controlled randomized stimulus distribution.

The last extra feature here is External Trigger. When the conductor wants to investigate the triggering operation (which itself will be described in the next section) without starting any session, he can send a trigger of any number to the recording device. In order to make this investigation easier, the software increments the trigger value sent, each time the External Trigger button is pressed. This value is also visible in the accompanying text boxes, therefore is easier to identify, comparing directly with the results in the recording software.

3.2.3 Triggering

As hinted before, a very important behind-the-scenes functionality of a stimulus software is its triggering capability, because knowing the exact time of the stimuli is the principal requirement for extracting information about them.

Our stimulus software has the ability to communicate with our recording hardware, BioSemi Active Two by utilizing the LPT port. Special care was taken due to the new Windows operating system regulations, as they forbid direct access to hardware resources such as the LPT port. The problem was handled by using extra DLL's that reach the LPT port in kernel level.

Once a session is recorded with a subject, it can be used in many different analyses and cross-analyses. It may also be shared with other research groups or people, who were not present during the actual recording session, for further analysis. With this possibility in mind, the triggering properties gain another significance, because the researcher that investigates the recordings should be able to master every detail. Moreover, for functional purposes and to have the easier hand at processing the recorded data in MATLAB, the triggering data should be as specific as possible. Therefore, our stimulus software sends additional informative triggering signals at significant times.

The principal trigger signals are positions of intensifications. Each row or column has a unique ID number, and whenever a row or a column is highlighted, corresponding ID is sent over the trigger channel to the recording device.

Rows and columns are numbered from 0 to $n_{rows} + n_{columns} - 1$, where rows are numbered from 0 to $n_{rows} - 1$ and columns are numbered from n_{rows} to $n_{rows} + n_{columns} - 1$. Therefore, when reading the recorded data file, the analyst has the possibility to extract the exact position of the intensification in the matrix, given that he knows the size of the matrix (see Chapter 4 for detailed information on how to acquire the basic information about the recording session, i.e. the dimensions of the matrix).

A tricky problem surfaces here, while trying to decipher the data. As we mentioned before, the stimuli are block-randomized, and the analysis software follows the trigger channel for discrete value jumps. Now, in a scenario where the last random element in a block, is the same as the first random element in the next block, although a new trigger value is sent, since the value remains the same, the analysis software is unable to detect the new trigger information. Therefore, a control over block randomization is required and implemented in this stage: the first random element in a block cannot be the same as the last random element of the previous block. If such a condition occurs, randomization is performed again; until such a problem is avoided.

For a healthy examination, these basic position trigger signals are not enough due to the aforementioned reasons and limitations. Therefore, the software features extra triggering information.

First, since the software that analyzes recorded data extracts trigger information from level differences in the continuous trigger data channel, to mark the beginning of a session, the software sends a value of 255 - the maximum value the trigger signal can have - a value only used for this first time initialization. After this initialization, the next trigger signal, any value other than 255, will be detected safely and accurately by the analysis software.

Additionally, at the beginning of each run, the software sends two more trigger signals that help the analysis software define a new run. The first one is the ASCII value of the target character, and the other one is again, the position of the target letter in the matrix. ASCII value of the target character is used for easier examination purposes, and the position of the target character helps to easily label the real trigger signals by matching them to these values.

For practical reasons, an important assumption is made here: any of the matrix dimensions is expected to be at most 9. With this rule, the position of the target letter is sent to the trigger channel by concatenating each dimension value and adding a 1 in front. For example, if the letter is A, 100 is transmitted. If the letter is C, 120 is transmitted. Therefore, the analysis software can obtain the targets by reading the second trigger signal, after the ASCII representation, and cropping out the leading 1 and

separating the digits. Of course, if need arises, an appropriate triggering protocol may be developed to handle larger dimensions.

The target letter and target position in the matrix, and most importantly knowing when a run ends and a new one starts are key elements to easily decipher the recorded data into recognizable data structures.

CHAPTER 4

OFFLINE ANALYSIS

This chapter will focus on the details of signal processing that occur, and will feature experimental results for offline analysis and comparisons to results of other BCI groups.

4.1 Background

Offline analysis of signals is used in conditions where the experiment is conducted independent of the analysis, either by necessity, or choice. The situation might even be that experiments are conducted in one laboratory in a broad period of time, and then the analysis is done in another place, with all the experimental data at once.

In P300 BCI context, the offline analysis finds use in developing classification algorithms and techniques that just need a data set to work on, as well as assessing and cross-validating known techniques. On the other

hand, offline analysis has limited use, because there is no feedback capability to indicate the classification result and show the subject his choice.

For evaluating and optimizing the performance of our analysis method later to be used in online analysis, we conducted several offline analyses with prerecorded data, both from EPFL BCI experiments, and from our own recordings. The methods and results will be presented in the following sections.

For classifying a data set, one needs a classification software. Since such a software will involve extensive calculations on matrices, a platform suitable for mathematical calculations is preferred. For developing our software, we chose MATLAB.

4.2 Classification Software

The classification software we have developed in MATLAB for offline analysis of the experimental data we obtained, reads the BDF (BioSemi Data File) file that holds the recorded data into MATLAB, creates the necessary data structures, and does the training and testing of the chosen classifier with the given data.

The basics of the analysis method are structured around the techniques used in [17], with differences that are essential for our purpose. As the data structures are formed after the recorded data file is read, data

epochs of 1-second periods that follow each trigger signal are extracted and each epoch is labelled as 1 or -1 according to the target stimuli.

We have conducted recordings for 8 letters - 8 runs - in a session for the purpose of this analysis. According to this, epochs are placed inside the data structures for each run. Please check section 4.3 for detailed information about the method.

We have used the Bayesian Linear Discriminant for the classifier. The algorithm is proposed in [22], and the actual code is developed by Ulrich Hoffmann of EPFL BCI group in 2006. The classifier uses the first of two sessions as the training set, and the other session is fed into the classifier as the test set.

The classification problem here is whether the epoch in question contains a P300 wave or not, or in other words, if the stimulus in question creates a P300 wave or not. According to this, it is assumed that the subject could successfully process the matrix and the given instructions in the first session, as it is used for training purposes.

The classification software investigates trials in the recorded data in groups of 12. A classification score is generated according to the distance of data in each epoch to the training set. These scores are separated for columns and rows and trials with maximum scores in each group are selected as the result for the classification process. Since this result includes

a row and a column, the intersection points to the letter (matrix element) the classification algorithm obtained from the analysis of the recorded data.

If this letter is the same with the letter instructed to the subject during the experiment - and therefore the same with the letter recorded in trigger signals - the classifier performed correctly, and we have a correct result.

4.3 Terminology

A target is the letter the subject is instructed to focus on at that instant.

A trial is the intensification of each row or column, and is denoted by **trigger signals** in the recording, which are recorded in a separate, special channel to define time markings for important events (such as trials, and markings of the beginning of each run, etc.). Trials are events that occur successively and rapidly.

Defining the dimensions of the matrix rows and columns as a and b , the trials are flashed in a block-randomized fashion, in groups of $a + b$. These groups are called **trial groups** and when all the elements in a trial group is flashed, a row or a column is flashed exactly once and there are no rows or columns that are not flashed, or flashed more than once.

With this in mind, a trial group is the smallest data set for a P300 classification problem.

A run is the collection of trial groups. A run is recorded for each letter defined in a session. There can be a period of a few seconds between each run, but the recording is not interfered with, and continues.

In each **session**, the P300 stimulus software is executed once and consecutive recording may be done for a desired number of target letters - runs. The conductor should allow a brief period of a few minutes between two successive sessions to allow the subject a period for resting and relaxing.

A session group is the collection of all sessions recorded with one subject during the course of a day, with delays in between each session.

To obtain meaningful results with P300 waves using the oddball paradigm, one has to provide a training set of at least one trial with a label to indicate a P300 wave, and another trial with a label to indicate otherwise, and a test set of at least one trial group.

To increase the performance of the classifier, the number of recorded trial groups in a run in the training set has to be increased. In addition, to obtain better and clearer results and to allow room for errors, the size of test set should be increased.

4.4 Method

4.4.1 Preliminaries

The method we have followed during the experiments is as follows:

We have two sessions in our session group. The rules and specifications for the first session are universal for each subject, and different for each subject for the second session. A 6×6 alphanumeric matrix of letters and numerals are standard in each session.

The target letters instructed in the first session are "D E D E D E D E" and therefore 8 runs are recorded with these targets. Before starting each run, the target is presented in the matrix to let the subject see where it is located in the matrix and therefore focus. As previously stated, according to the principles of our classifier, increasing the number of trial groups in each run increases the performance of the classifier and results in quicker and more correct classification. For this purpose, each run features 20 trial groups. Due to the size of the matrix, each trial group features 12 trials, 2 of which are targets and 10 of which are irrelevant stimuli.

Each trial lasts for 300 ms. For the first 125 ms, relevant row or column is intensified, and for the remaining 175 ms, all the elements in the matrix dim and the system waits for the next trial.

According to this, a trial group lasts for 3.6 s. Considering this duration, if period between each run is ignored and with the assumption of highest performance, 16 letters per minute is the limit of the performance. For various reasons, performance of classification processes that feature one trial group is not satisfactory. In contrast, using more trial groups in each run decreases the number of letters classified per second, while increasing the probability of correct classification. As stated in Chapter 2, state-of-the-art speed is about 6-8 letters / minute. With this research, one of our purposes is to propose a new stimulus method and increase the classification performance.

4.4.2 Data pre-processing

Recordings are done in conjunction with the method presented above, using the BioSemi Active Two EEG recording device. The software for recording the data is the native interface for the hardware, BioSemi ActiView. The data is sampled at 2 kHz. We use 12 active electrodes in the recordings which are placed in Fp1, Fp2, Fz, Cz, Pz, Oz, P3, P4, PO7 and PO8 locations according to the international 10-20 system, as well as two auxiliary electrodes for reference that are located on the mastoid channels.

ActiView software saves the recordings to the disk as a BDF file, whose format is developed by BioSemi. A BDF file has a similar structure to

an EDF file, but while the resolution of the EDF file is 16-bits, a BDF file has a 24-bit resolution. Although other specifications of the file format is the same as EDF, this resolution depth allows saving of more sensitive information.

The recordings saved in a BDF file is read into MATLAB via the code developed by Alois Schloegl in 1998. After this process, the raw data has to be structured and prepared for classification.

The trigger channel is extracted as the first part of these preparations. Times (sample numbers in sequence) and values (actual trigger values) of each trigger signal are obtained from the data in the trigger channel and stored in a key - value pair. This information will later be used for extracting epochs.

For a healthier analysis, the data have to be filtered. Especially to reduce the size of the feature space and rid the data of irrelevant frequency components, background noise and DC offset that occur between electrodes and the skin due to sweating, all the data are filtered with a 3^{rd} order Butterworth bandpass filter of a pass band of 1-12 Hz. This filtering removes most of the unwanted artifacts.

ActiView saves the data unreferenced, that is, referenced to CMS, the common mode sense electrode which is just another electrode attached to the head in parieto-occipital region. To allow for a greater SNR, the data has to be re-referenced to a channel or a combination of channels. The data

obtained from mastoid channels are assumed to contain body potentials due to muscle movement and no EEG signals, albeit relevant to P300 waves. Therefore, by taking the mean of two mastoid channels, a reference signal is acquired and by subtracting these from the other channels, the data are referenced.

Independent of any experimental procedure, this is the most fundamental data-preparation for recorded EEG data; in other words, a standard EEG recording has to be prepared in this fashion. Later on, more preparations specific to an experiment in question, might be necessary.

As the next step, we extract the ASCII coded letter trigger signals in the trigger channel. The data cell that holds the runs in a session is sized according to the number of extracted ASCII codes, therefore to the number of letters in the recording.

Later, each run is separated according to when the aforementioned triggers came and a standard data structure is formed for each run. A standard data structure for a run incorporates several parameters, such as a 'target' parameter that holds which letter (or symbol) the target is, a 'targetposition' array that shows where the target is located in the matrix by rows and columns, a 'stimuli' array that holds the trigger values extracted from the trigger channel, a 'labels' array for deciding if the elements in 'stimuli' array correspond with the target or not, a 'times' array that holds the sample times (sample numbers in sequence) of when an

element in the ‘stimuli’ array has occurred (therefore, the aforementioned key-value pair is obtained with the ‘times’-‘stimuli’ array pair).

The next preparation step forms the data epochs regarding the values in the ‘times’ array. Each epoch is a data set of 1 seconds. Since the data are digitized at 2 kHz, each epoch holds 2048 samples. To reduce the size of the feature space and remove the unnecessary features, the data are decimated by 64.

In the next step, to remove the negative effects of electrode-skin resistance that result in amplitude changes and other anomalies, the data are normalized.

Unfortunately, if the waveform involves very high, extreme values, normalization may result in a poor performance. To avoid this problem, the data are windsorized in a 10% window. Windsorizing the data removes the extremities, clipping the samples that are out of this window and gives a meaning to normalization. Normalization and windsorization classes are also developed by Ulrich Hoffmann. Experimental results show that, while windsorizing and then normalizing data works well with recordings of some subjects, others give better results without windsorization and normalization. Because of this fact, the method that gives better results were chosen for each subject. This detail is expressed in figures in section 4.5.2.

With these last steps, the data are ready for classification. In the final state, an epoch is represented as an $m \times n \times t$ matrix where m is the number of samples per epoch (32 in this case), n is the number of channels (10 in this case) and t (generally 240 for 20 trial groups) is the number of epochs.

4.4.3 Classification

The feature vector for each epoch is then the concatenation of filtered data from each electrode, e.g. a vector of 320 samples for 10 electrodes. In the end, for the classification algorithm to work on, the data are reshaped as a matrix of size $r \times t$ where r is the size of the feature vector (320 samples), and t is the number of feature vectors.

As mentioned before, the classifier algorithm is chosen to be Bayesian LDA. The training set is the first session, whereas the test set is the second session, partly, or in full. Bayesian LDA calculates a score for each element (epoch) in the test set reflecting the distance defined by the training set. After that, the scores for epochs grouped in sizes of a trial group will be calculated. Since the matrix in question is 6×6 , the epochs are handled in groups of 12.

For example, if a complete run is used as a test set, according to the method explained above, there are 240 epochs in it. A total of 20 result

pairs are generated and rows and columns with the highest score are obtained as results. If the position of the target corresponds with these values, the classifier results are registered as correct.

4.5 Experiments and Results

4.5.1 Experiments

For offline analysis in our research, we have made experiments and sessions with 4 male and 5 female able-bodied subjects whose ages range in between 19 and 25.

At least two sessions were recorded with each subject, whose first session almost always consisted of "D E D E D E D E" and the second session consisted of random letters.

For the results shown below, the following approach has been pursued: All the epochs in the first session are used as the training set, and the second session is used as the test set. Plots feature two Y axes, where the one on the left shows the accuracy of the classifier against time, and the one on the right shows the bit rate of the classifier against time. We have previously stated that a trial group lasts for 3.6 s. Therefore, a run made of 20 trial groups lasts for 72 s. Intuitively, the classifier performance is expected to increase as more test data are acquired, i.e. more time passes.

The definition of the bit rate is mentioned in [41], first introduced in [34]. Basically, the bit rate shows the rate of information that would be transmitted per minute with the given accuracy. Defining P as the accuracy of correct classification, N as the number of elements in the matrix, and wrong classification accuracy as $(1-P)/(N-1)$, then bits/situation is

$$B = \log_2 N + P\log_2 P + (1-P)\log_2[(1-P)/(N-1)] \text{ where } P \geq \frac{1}{N}$$

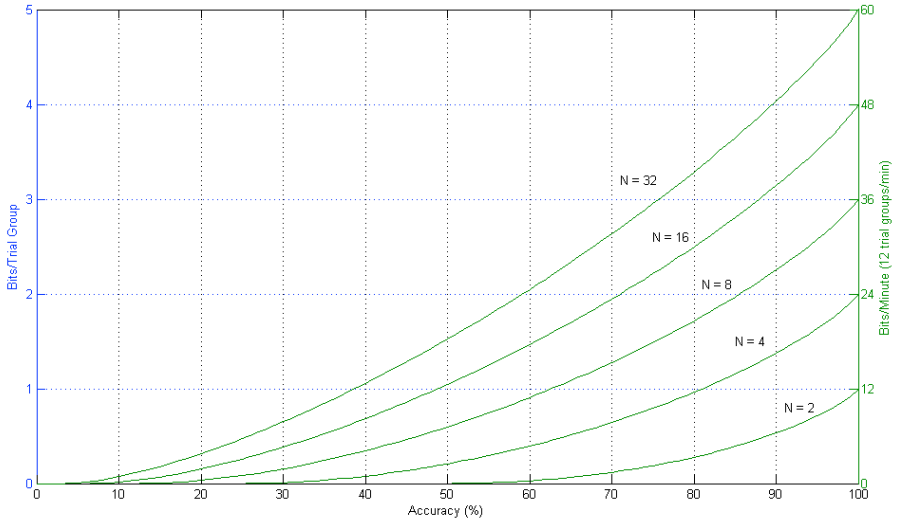


Figure 4.1 Bit rate plot versus Accuracy (%) for varying N's. One trial group of our system lasts for 3.6 s. Adding another 1.4 s for feedback purposes, an average trial group takes 5 s. Therefore there can be 12 trial groups in a minute.

For our analysis, the number of choices is 36. Therefore, in a trial group, we code $\log_2 36 \cong 5.1699$ bits/trial group. Assuming a trial group lasts for 5s (see Figure 4.1), maximum bit rate of our system is 62.0391 bits/minute on the average.

For calculating accuracy, every run is classified separately and the accuracy is the total number of correct guesses in a session over total number of guesses. Since there are 8 runs in a session in general (see results for exceptions), there are 16 correct guesses at most. This means that 100% accuracy is achieved if the classifier guesses all the 16 results correctly.

4.5.2 Results

We show the classification results of 9 subjects in Figures 4.2 through 4.9. The results are presented in a format directly compatible with [17]. It is easily seen that subject 1 and 2 performs best. Talking about the subject's experience after each session, subject 1 and 2 stated that their concentration were utmost. Subject 3 stated that she might have counted a wrong letter in a run. Although she could not remember the exact run of the wrong letter, a corresponding result in Figure 4.4 shows that this is actually the case. The classifier has a 100% accuracy from 2^{nd} trial group (which is a really impressive result) to 7^{th} trial group in all runs. In contrary to the general expectation about the classifier performance stated in the previous section,

in Figure 4.4, after 7 runs, we see 1 wrong conclusion (93.75% accuracy) until the end of the runs. This might be due to wrong focusing of subject 3.

For subject 4 and 5, a run consisted of 10 trial groups instead of 20, and instead of mastoid channel references, averages of all 10 electrodes are used. The electrodes are consistent with other subjects. As a deviation, subject 4 used 8 totally random letters for the training set, and chose the word 'MOZAIC' for the test set. Subject 5 used the standard 'D E D E D E D E' for training set and wrote 6 random letters as the test set.

Although two sessions were recorded with Subject 7, due to a file handling error the first session's recordings were lost. For Figure 4.8, other subjects' recordings were tried as the test set for the classifier, and the best performing one was chosen (second session of Subject 6). Figure 4.8 is gives an important intuition about intra-subject classification performance. Although the performance is not optimal, the 100% accuracy achieved shows that, in contrast to other fields of BCI communication, P300 speller paradigm can tolerate subject variability.

Subject 8 reported to have concentration problems. The varying performance reflected in Figure 4.9 confirms this. Some runs are classified correctly and some runs have only one correct classification result.

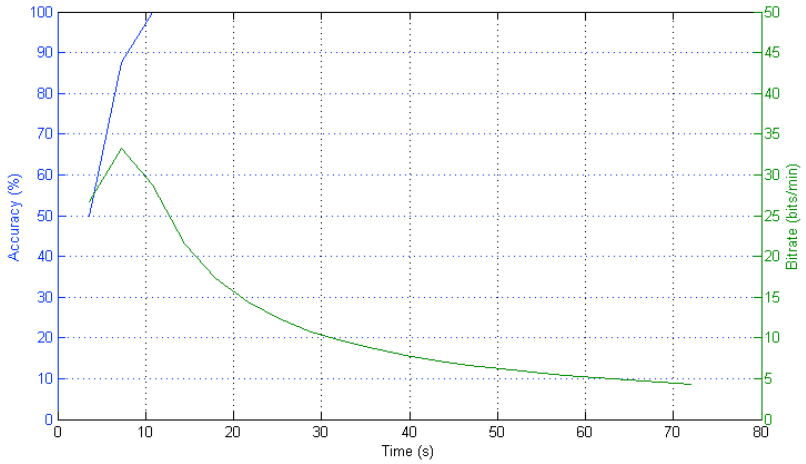


Figure 4.2 Offline analysis results for subject 1. No windsorization and normalization applied during data preparation.

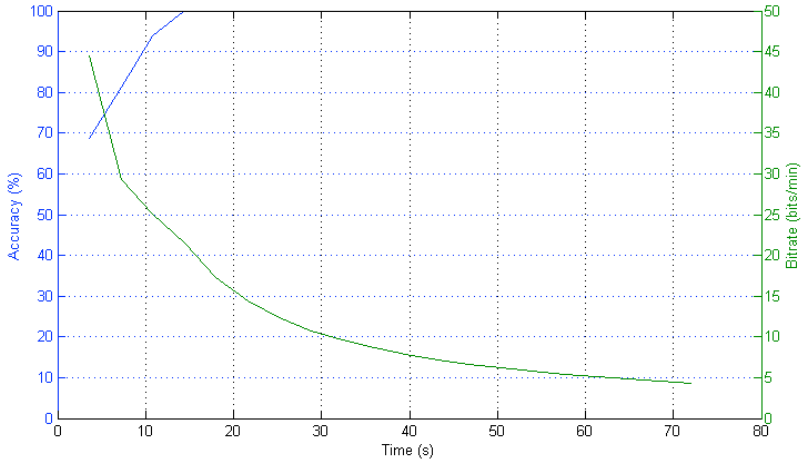


Figure 4.3 - Offline analysis results for subject 2. No windsorization and normalization applied during data preparation.

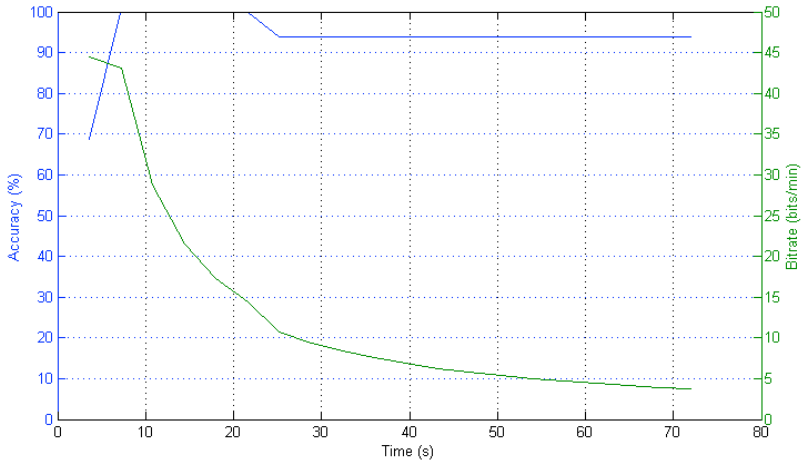


Figure 4.4 Offline analysis results for subject 3. No windsorization and normalization applied during data preparation.

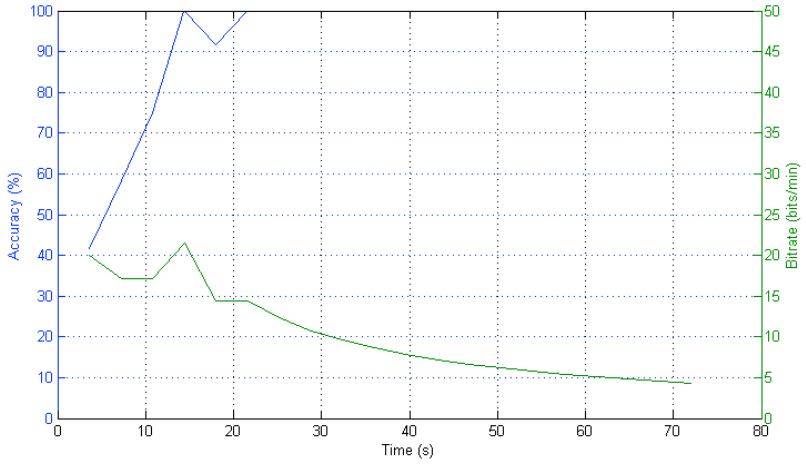


Figure 4.5 Offline analysis results for subject 4. Windsorization and normalization applied during data preparation.

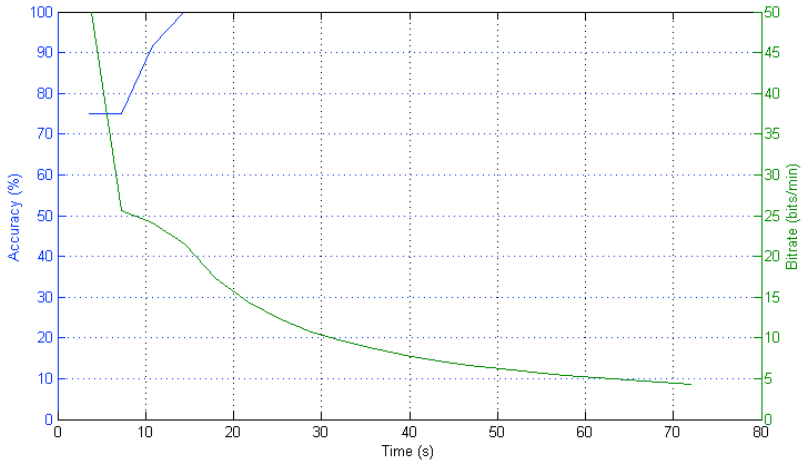


Figure 4.6 Offline analysis results for subject 5. Applying or removing windsorization and normalization yielded same results.

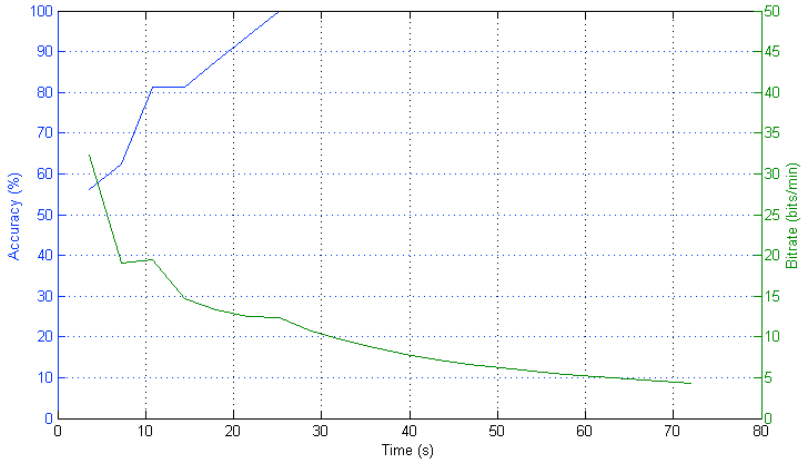


Figure 4.7 Offline analysis results for subject 6. No windsorization and normalization applied during data preparation.

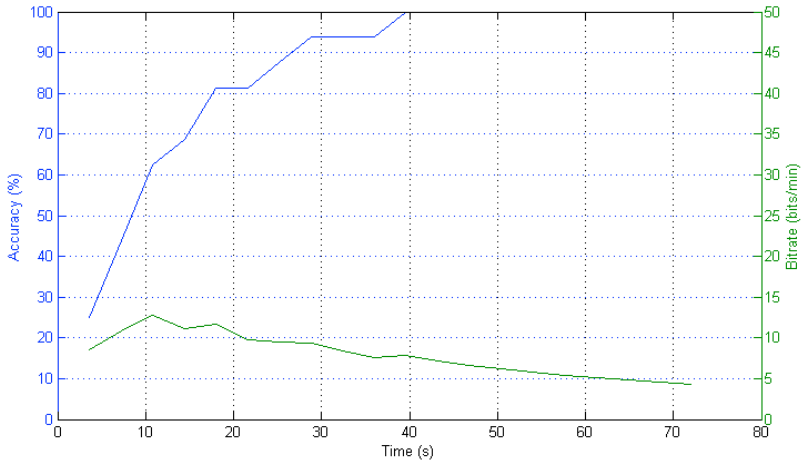


Figure 4.8 Offline analysis results for subject 7. Windsorization and normalization applied during data preparation.

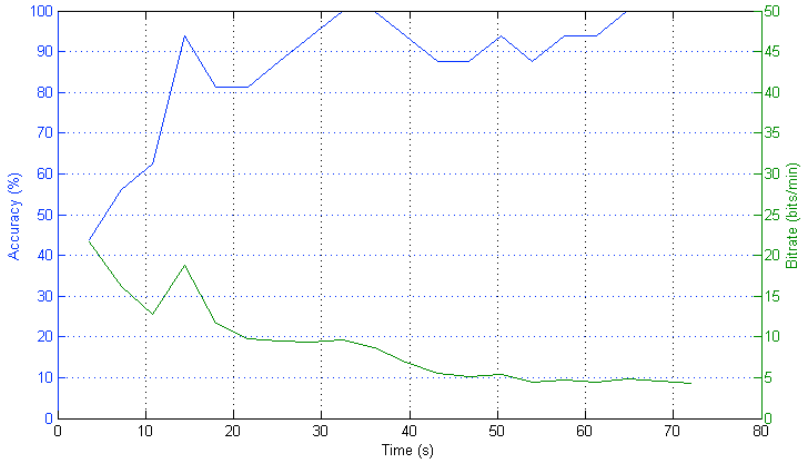


Figure 4.9 Offline analysis results for subject 8. Windsorization and normalization applied during data preparation.

4.5.3 Discussion

Figure 4.10 shows the average classification performance. The x axis shows the number of trial groups, and the y axis shows the percentage the classifier predicted correct results. This figure tells us that in the ideal case with one trial group, 58% of the time the classifier predicts the right answer. For the rest of the time, the classifier has correct answers in 2 trial groups 74% of the time and so on.

This result suggests that for 89% of the time, a letter can be classified correctly in only 2 trials. Considering the information in Figure 4.11, on average, a given set of 100 runs will last for 154 trial groups. This in turn shows that each run is on average classified in 1.54 trial groups. Assuming a period of 3.6 s for a trial group, this implies that a letter can be classified in 5.544 s on average.

If we assume no delay between each run, then on the average our subjects can write 10.822 letters / minute. Unfortunately, one has to give a break to the execution of the experiment since feedback for a new letter is necessary to let the subject focus on the letter. If this interval between each letter is a brief 1.4 s as mentioned in Figure 4.1, the average letters per minute, found as a result of experiments done with 8 subjects, is found to be 8.6405.

This performance versus trials needed for 100% classification is illustrated in Figure 4.11 and Figure 4.12.

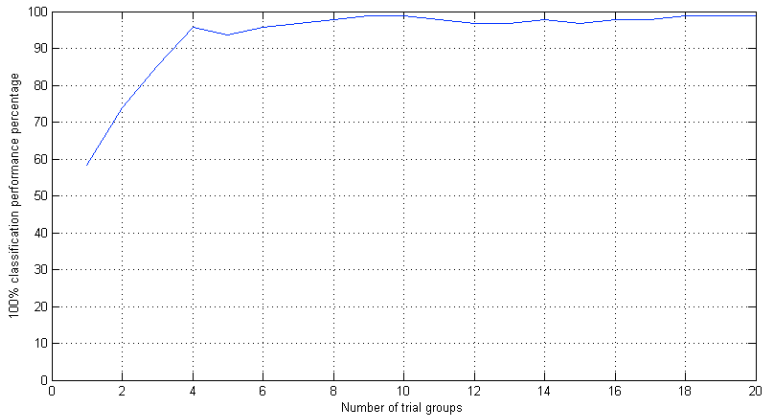


Figure 4.10 Classification performance averaged over 8 subjects. The figure suggests that in 4 trial groups, more than 95% of the time the classifier predicted the correct target.

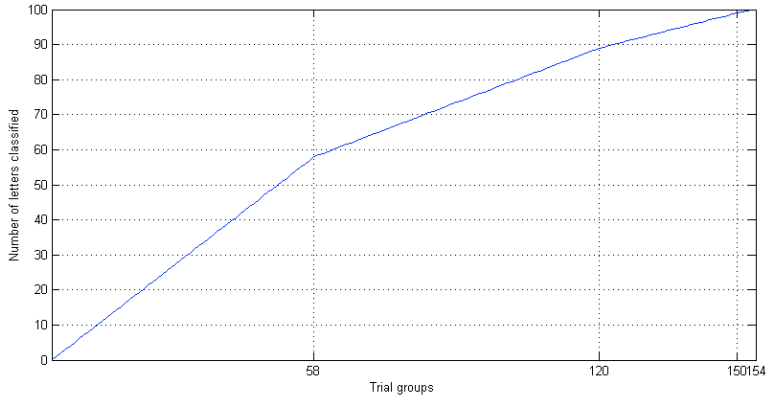


Figure 4.11 Distribution of a sample of a 100 letters, showing average classification performance of our subjects. The easiest 58 trials are classified with an accuracy of 100% in the first trial group, the next harder 31 trials need 2 trial groups, the next harder 10 trials need 3 trial groups and the hardest trial take 4 trial groups.

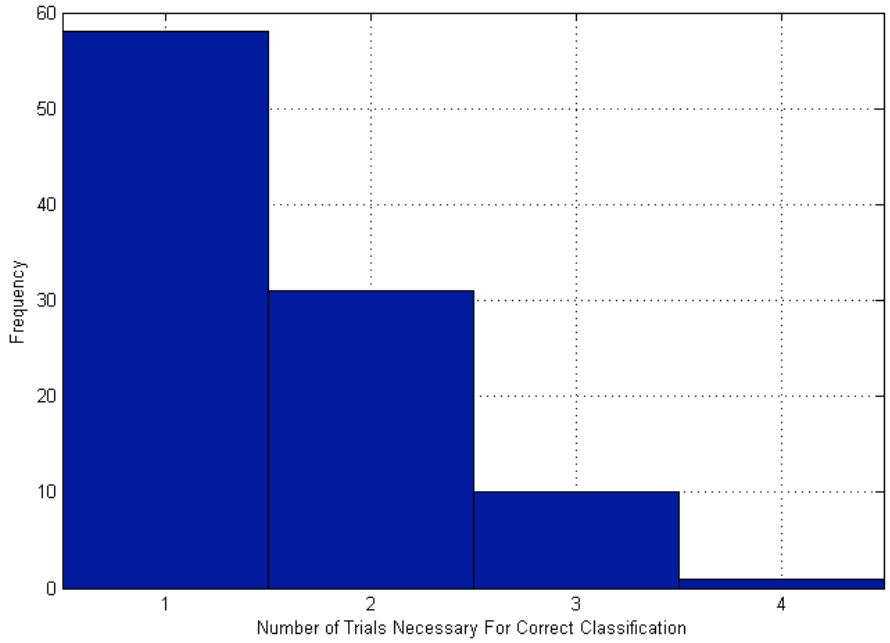


Figure 4.12 Histogram of data in Figure 4.11, showing how many letters take how many trial groups for correct classification.

CHAPTER 5

ONLINE ANALYSIS

This chapter focuses on online recording, analysis and feedback of P300 signals. In chapter 4, we explored the capabilities of offline analysis, where the brainwaves are recorded first and the analysis is done separately afterwards, allowing one to do various kinds of analyses on the data. In online analysis, however, brainwaves are analyzed concurrently, as the recording continues. We have proposed and implemented improvements in our algorithms and software for online operation and efficiency. We have demonstrated reliable real-time operation of our system. On average, our system achieves 12.48 letters / minute with 89.75% and 11.14 letters / minute when 100% task completion is ensured This means the subject types all the letters he/she wants. There still might be errors in results, but they are compensated by the subject by retyping the wrong letter. The time that is lost due to retyping is taken into consideration in this result. Therefore we might say this is a pseudo-100% accuracy result.

5.1 Background

There are two types of online analysis used in P300 speller research, according to the experimenter's prior knowledge about the targets during the session. In one, the experimenter selects the target letters and gives them to the stimulus software to have the subject focus on those during the session. Since conductor knows what targets to expect and therefore the stimulus software records the trigger signals of these letters, the evaluation of system performance may be done with ease. Of course, this type of analysis is not really suitable for practical use, as the purpose of the whole speller is to provide disabled subjects a medium for interaction where they are free to write whatever they want. Nevertheless, in terms of evaluation of system performance, this method is frequently used since it allows more sophisticated analyses to be done off air, after the recording of the session is complete, just like offline analysis, using the knowledge of epoch labels. The difference with offline analysis is, the data are filtered and processed in real-time, as they arrive. In other words, the data is first divided into epochs and then filtered and processed locally in epochs, instead of filtering and processing as a whole and then dividing into epochs. There are several issues that arise due to this in-epoch processing, and these will be investigated in section 5.2. In this type of analysis, runs always consist of a predetermined

number of trial groups (see section 4.3 for trial group). In other words, the number of trial groups in a run is not dependent on performance.

The second type is where the subject is completely free to choose the target letter to focus on. This time, the conductor has no prior knowledge of the target, and since there are no labels recorded with the data, he has to find it out only by classifying and evaluating the incoming signals. In this type of analysis, runs usually consist of different numbers of trial groups. The reason for this is as following:

Since this type offers no labels for epochs, the experimenter is unaware of the real target of the subject and has to find it out by analyzing the signals. Although waiting for a predetermined number of runs, and then looking at the classification results is possible, there usually is no need, because performance of the subject changes on every run; in one run, there might be good classification accuracy in 3 trial groups, and in another run, the same accuracy might be achieved in 6 trial groups. Now in the case that there is a predetermined number of runs, either one accepts false classification by not waiting enough for 6 trial groups, or waits redundantly for the answer might be already ready as soon as 3 trial groups. Making the system in a way to support random number of trial groups per run is a good idea to handle these problems.

Therefore we have developed a greedier version of our algorithm that relies on the fact that the classifier produces probabilistic scores. In the

beginning of each run, each row and column receives a score of 0. If their epoch includes a P300 wave, they get a positive score, with its magnitude reflecting the resemblance to the training set, and irrelevant epochs get a negative score. With this approach, there is usually no need to evaluate all the 12 epochs for a decision. If the score of an epoch already satisfies the margin, the decision can already be made.

Our analysis is based on this type, as it reflects the real usability of the system. From now on throughout this thesis, this type will be referred to as simply "online analysis".

5.2 Problems and Observations

Evidence shows that background activity blurs the P300 activity in a quite random fashion. Therefore, averaging the incoming data for a better signal to noise ratio and easier analysis yields better results. In averaging, there are rows + columns number of groups, and members of each group are iteratively averaged (e.g., epoch of the first row in the matrix, in a trial group, has to be averaged again with the epoch of the first row in the next trial group). This is a huge disadvantage when one needs quick results, as the purpose of studies in this field is to decrease the number of necessary trial groups for correct classification, where the ideal number is 1. Clearly the second best performance, a correct answer per two trials, is not long

enough for averaging. In addition, in the previous chapter, we have shown that a large percentage of targets are classified in a few trials. This fact makes averaging unsuitable for our purposes.

Another slight disadvantage in online analysis is in bandpass filtering. In offline analysis, bandpass filtering is done on a whole recorded data in a channel; in other words, the data to be filtered include a complete run. On the other hand, in online analysis, one has to filter and process data as they arrive in epochs. It is easily seen that epochs are just extracts of a recorded data of a run. Filtering parts of a data separately instead of filtering the data in whole produces a problem in dealing with low frequency components, as they are reflected in the recording as offset drifts. Filtering the data as a whole, helps one to have more samples, therefore even the smallest frequency component may complete a period, whereas in filtering by epochs, one cannot decide whether there is a sub-Hertz component or not.

Figure 5.1-3 show the same epoch of data in three different conditions; unfiltered, bandpass filtered before epoch extraction and bandpass filtered after epoch extraction. Figure 5.4 has the latter two on the same axis, to show the difference. Figure 5.5 shows the power spectrum of the same epoch in Figure 5.1-4. It is clearly seen that in-epoch filtering has a suppressive effect on low frequency components.

Keeping in mind that ISI is 300 ms (epochs have 300 ms delay between each other) and each epoch lasts for 1000 ms, it is clear that two consecutive epochs overlap. The last 700 ms of an epoch is the same as the first 700 ms of the following epoch. With this in mind, the error that occurs in in-epoch filtering can be visualized as in Figure 5.6-9. The second epoch (green line) is delayed to match the corresponding data points with the first epoch (blue line). For easier inspection, Figure 5.6 shows the two epochs separated on the same axis, regardless of their amplitude values. Figure 5.7 shows the two epochs overlapping. Note that the last 700 ms of the first epoch overlaps exactly with the first 700 ms of the second epoch. Figure 5.8 shows the two epochs separately, this time in-epoch filtered. Figure 5.9 shows two epochs overlapping. Notice that, either one of the epochs is wrong about estimating the actual data, i.e., the filtering errs.

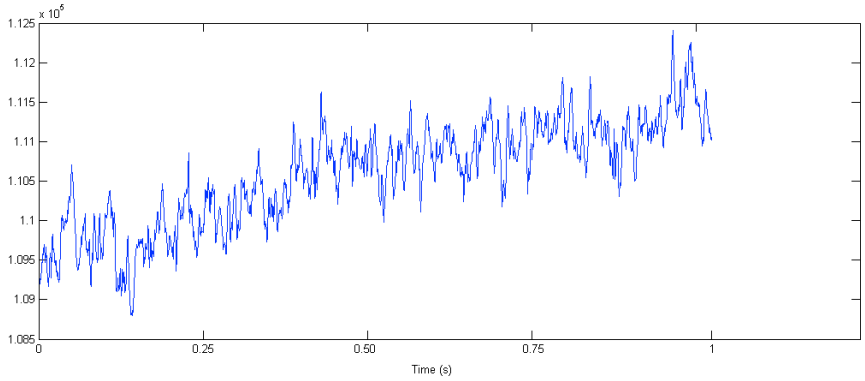


Figure 5.1. Raw epoch data

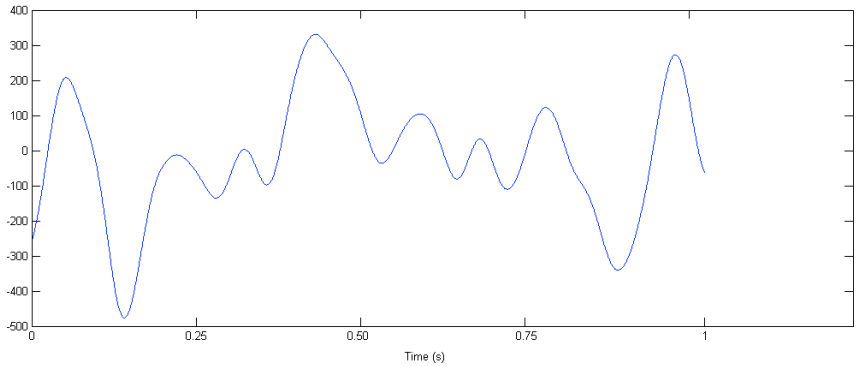


Figure 5.2. Epoch data filtered before epoch extraction

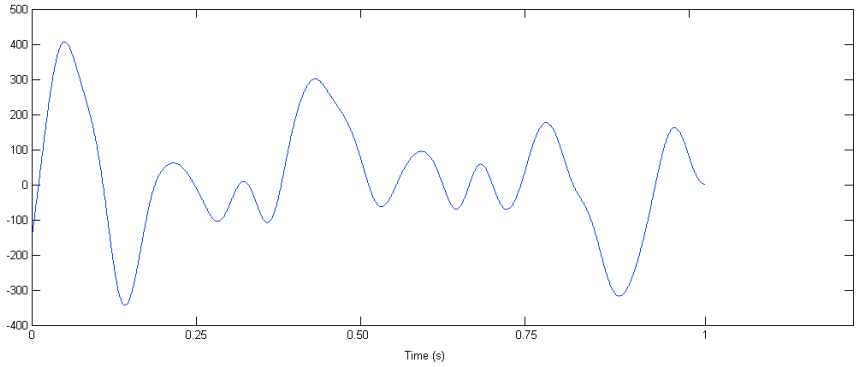


Figure 5.3. In-epoch filtered data

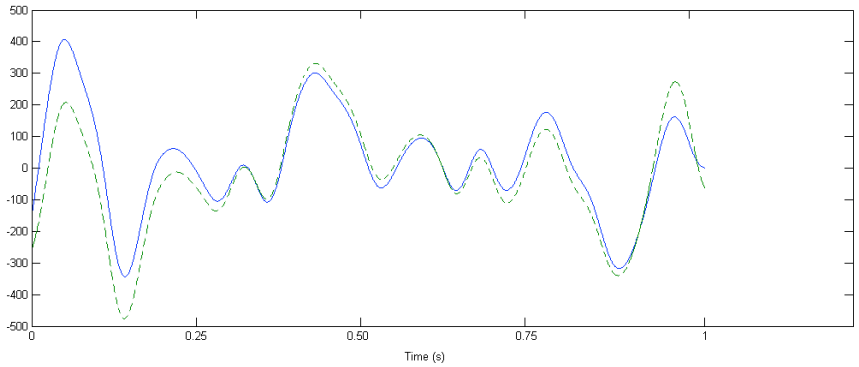


Figure 5.4. Two different filtering schemes in one plot. Dashed line is data filtered before epoch extraction, solid line is in-epoch filtered data.

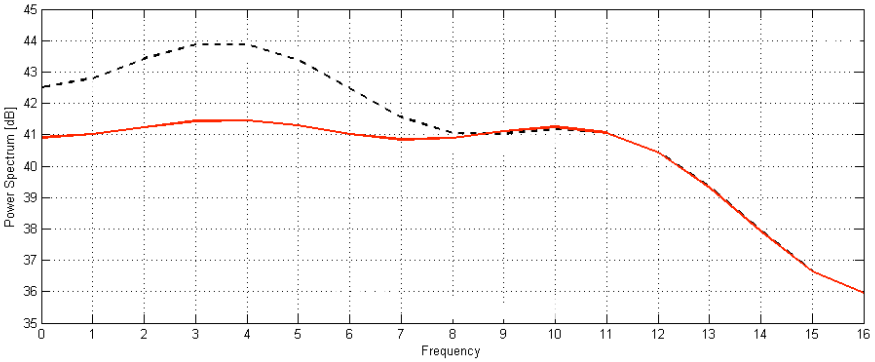


Figure 5.5. Power spectrum of the epoch in Figure 5.1-4. Dashed line is spectrum of data filtered before epoch extraction, solid line is spectrum of in-epoch filtered data.

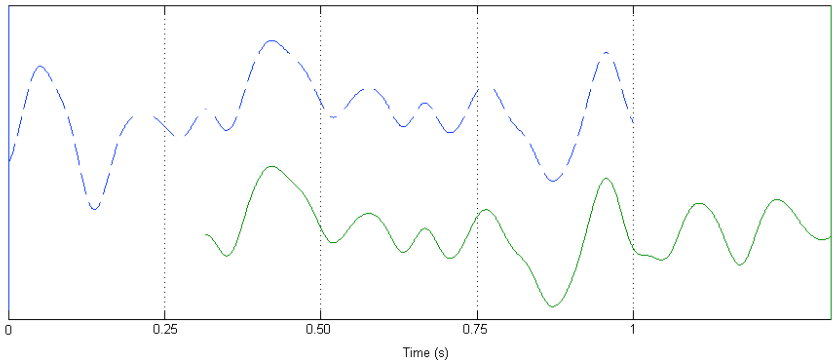


Figure 5.6. Two consecutive epochs whose data were filtered before epoch extraction. The latter one (solid) is delayed for approximately 640 samples to overlap with the first one.

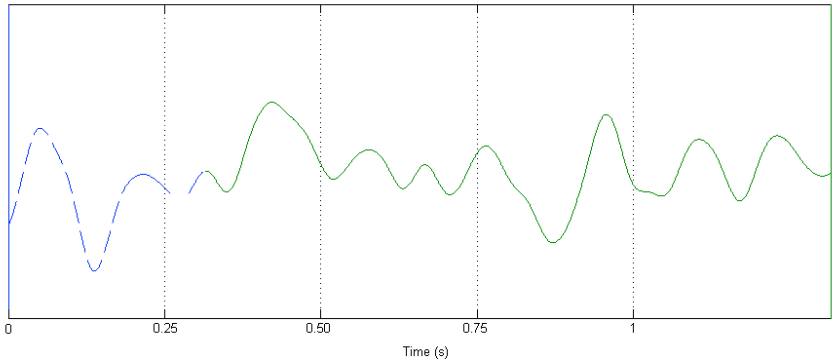


Figure 5.7. Two consecutive epochs whose data were filtered before epoch extraction. Note how they perfectly overlap, because essentially they are consecutive parts out of the same filtered data

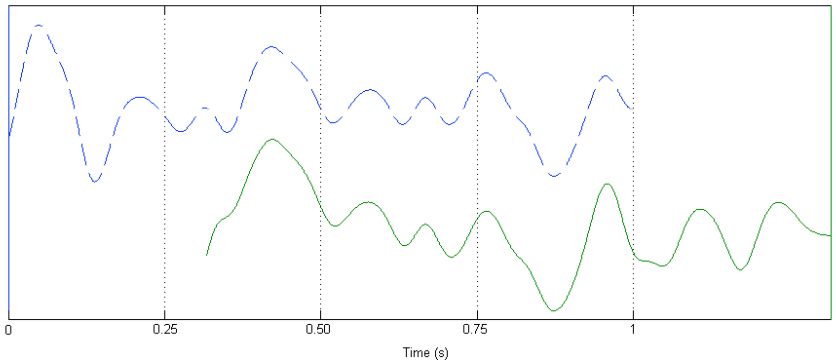


Figure 5.8. Two in-epoch filtered consecutive epochs.

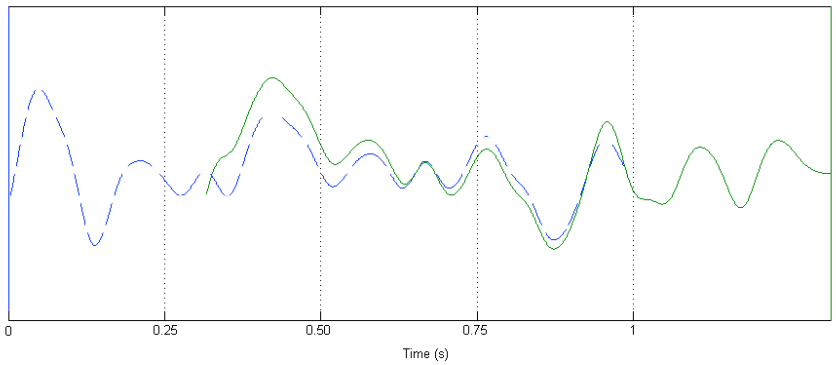


Figure 5.9. Two in-epoch filtered consecutive epochs. If there were no filtering errors, the epochs would overlap perfectly, as in Figure 5.6

Figure 5.10 shows a standard classification performance of an offline analysis (where data is filtered before epoch extraction - no windsorization and normalization applied) and Figure 5.10 shows the results of the same analysis where data is filtered in epochs (without windsorization and normalization again). A similar performance drop like this one is observed in

most of the subjects, where a few exceptions exist. There were 4 letters in question in Figure 5.10 and 5.11. Correct classification of all letters in Figure 5.10 lasts 8 trial groups, 2 trial groups on average: 7.2s/letter. On the other hand, correct classification of all letters in Figure 5.11 lasts 11 trial groups, 2.75 trial groups on average: 9.9s/letter. These results tell us that, the order of filtering and epoch extraction has an impact on classification. Although we are free to choose the best order for offline analysis, for online analysis, filtering has to be done in-epoch. We conclude that we are forced to accept a worse performance, with actually the same data.

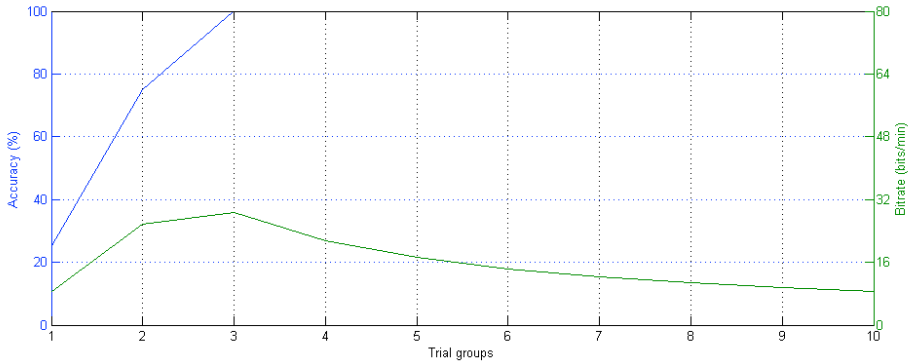


Figure 5.10. Classification performance of wholly filtered data

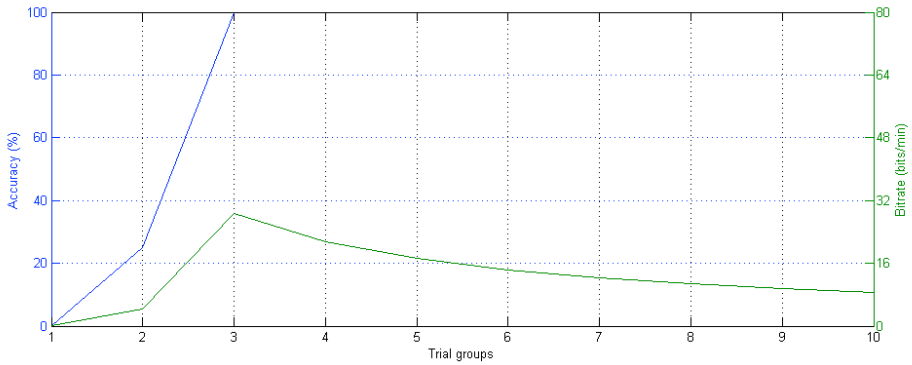


Figure 5.11. Classification performance of in-epoch filtered data

Yet another problem is in windsorization and normalization. Practically, windsorization cuts the negative and positive peaks that lie in a margin of the maximum and minimum values in the given data. If one applies windsorization to data of a complete run, some smaller peaks will not be windsorized, in other words, some epochs might have no values that lie in the window. On the other hand, windsorization in each epoch has little effect on peaks, because now the window will include less samples (due to the fact that an epoch is very short compared to a total run). Also, it might remove valuable data points in the case that there are no extreme points or peaks in that epoch. Figure 5.12 shows an epoch of data windsorized before epoch extraction, and Figure 5.13 shows the same epoch of data locally windsorized. Note that this epoch includes a peak. Figure 5.14 and 5.15 show another epoch with different windsorization schemes, but

in this case there are no such peaks. There is a 10% window in Figures 5.12-15.

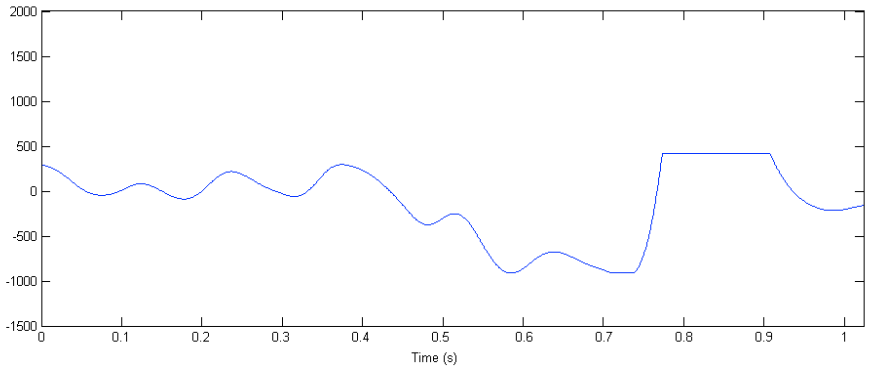


Figure 5.12. Data windsorized before epoch extraction, includes peak

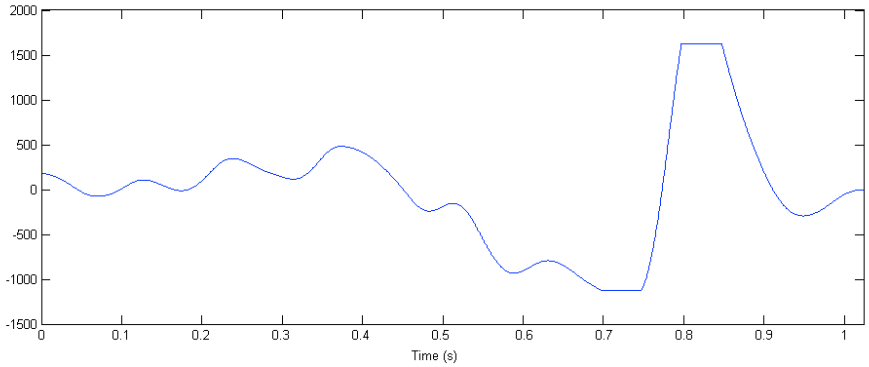


Figure 5.13. Data in-epoch windsorized, includes peak

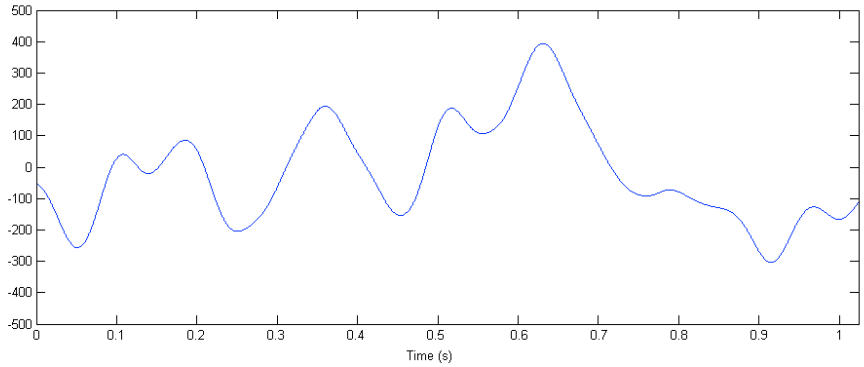


Figure 5.14. Data windsorized before epoch extraction, no peaks

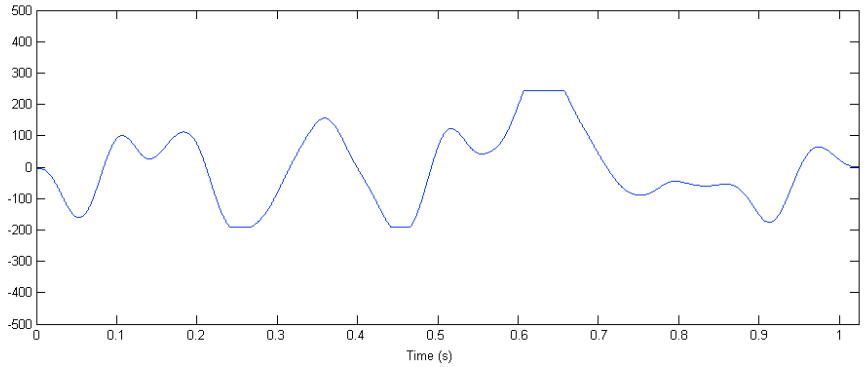


Figure 5.15. Data in-epoch windsorized. Notice that the small, local peak is unnecessarily windsorized.

Unfortunately, normalization presents a similar situation, though it is not always a problem. Normalizing a data with peaks will suffer from low resolution, because to represent a peak in a normalized data one has to squeeze smaller-valued data points. We use zero-mean normalization, where the mean of the given data points are matched to zero. Although this

approach provides better resolution in small-valued data points, it is still prone to resolution loss. Windsorization helps normalization here, by removing some of those peaks. Figure 5.16 shows the effect of global and local normalization on the same epoch of data (no windsorization applied in both). Blue line represents global normalization, green line represents in-epoch normalization. Clearly, global normalization has better resolution. Figure 5.17 shows the same epoch with windsorization applied. Note that, again, resolution is better in globally processed data and windsorization increases resolution.

Evaluating all of these different options to get a decisive scheme that performs good on all subjects showed us that it is practically impossible, and these nuances are subject dependent. Although we have shown with figures that processing data as a whole before epoch extraction looks good, classification performance vary from subject to subject. Performance of 6 out of 8 subjects dropped with in-epoch processing (as exemplified in Figure 5.10 & 5.11), whereas performance of the remaining 2 subtly increased.

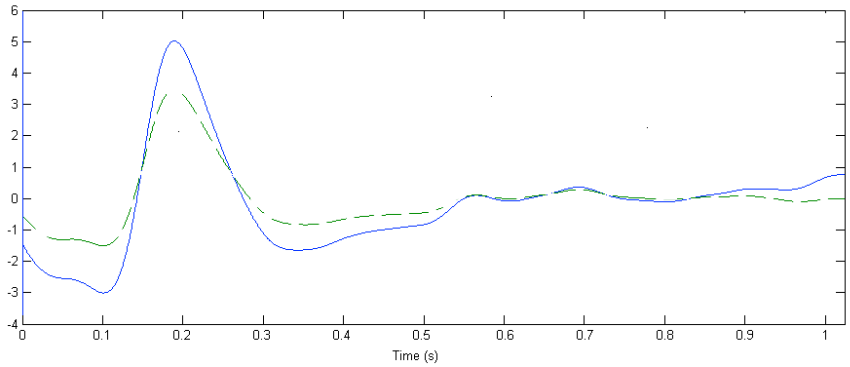


Figure 5.16. Different normalization schemes on the same epoch. Solid line represents global normalization, dashed line represents in-epoch normalization

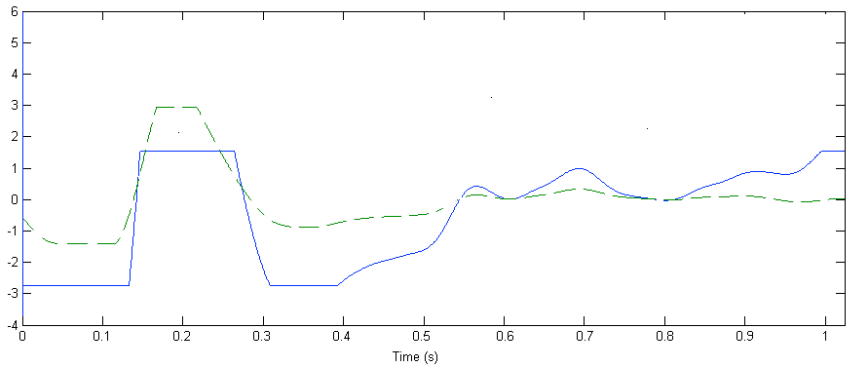


Figure 5.17. Windsorization and normalization applied to the same epoch. Blue line represents global processing, green line represents in-epoch processing.

5.3 Method

5.3.1 Preliminaries

To compare our results to other published works, we used two kinds of stimulus paradigms in online analysis, as in offline analysis. One is the classical gray and white matrix of 6x6 size that features letters and numerals.

On top of this, we propose another paradigm, where the letters flash in a randomly colored fashion. We believe this stimulation technique will fire a bigger P300 response in the subject's brain, because the subject might get used to white flashes and expectations might arise. In a randomly colored fashion, there are two surprising events; one, as usual, the subject is unaware of when the flash will happen, and two, the subject is unaware of in what color the flash will happen.

Figure 5.18 shows an example where the same column is highlighted in different colors at different times.

(a) (b)

Figure 5.18. Same row highlighted at different times.

The algorithm for random color generation is designed to select a random color out of six colors whose RGB values are listed in Table 5.1. Basically, each combination of 0 and 255 for each color is considered, and two colors are removed; black (since the background is also black), and due to subjects' comments, blue.

Color Name	R	G	B
Green	0	255	0
Aqua	0	255	255
White	255	255	255
Yellow	255	255	0
Red	255	0	0
Magenta	255	0	255

Table 5.1. Details of colors used in random-colored stimulation paradigm

For online analysis, we have different numbers of sessions in our session group for each subject, where the first session is always the training session. Instead of using letters "D E D E D E D E" like offline analyses, we have used 8 random letters for each subject. Each run features 20 trial groups, and each trial lasts 300 ms.

5.3.2 Data pre-processing

As we have mentioned before, the data are processed in-epoch. That is, incoming data are divided into relevant epochs first. After enough data to fill an epoch is streamed in, that epoch is ready for data preparation.

In the first step, the data are bandpass filtered in a 1-12 Hz passband to remove unwanted frequency components. Next, they are re-referenced to the mean of two mastoid channels. The data are then windsorized in a 10% window and normalized, and decimated by 64.

For details of this preparation, please see section 4.4.2.

5.3.3 Classification

As mentioned before, we use a greedy algorithm that gives faster results than the standard offline analysis method. This algorithm uses a fixed margin for all subjects, an empirical value decided by experimenting with data at hand. Observation shows that rows generally get smaller scores than columns. Therefore row margins are set at a lower value. When the two scores satisfy their margin, the intersection is displayed as the result of classification. Since the P300 response in subjects differ for all target epochs, scores vary, so the time to make a decision varies greatly.

5.4 Results and Discussion

Online performance values with color matrix are listed in Table 5.2. Note that these subjects do not necessarily follow the same numbering with subjects in Chapter 4. For easier comparison, we have included the offline analysis performance of subjects.

5.4.1 Results

Table 5.2 presents average online performance in detail, listing Right and Wrong classification results and two kinds of accuracy versus rate values. The first one allows errors in results, so typing rate is calculated disregarding the error in classification; therefore it is faster. In other words, the classifier produces wrong results, and these lower the overall accuracy for that subject.

	Average Offline Perform. (letters/ minute)	Average Online Performance (Color matrix)					
		Right	Wrong	Time (s)	Avg. Rate		Average Rate (lts/min exc. W)
					Acc (%)	Rate (lts/min)	
Subject 1	8.33	43	8	322.2	84.31	9.5	8.01
Subject 2	9.52	16	2	69.6	88.88	15.52	13.79
Subject 3	7.80	9	0	52.2	100	10.34	10.34
Subject 4	7.69	9	1	41.4	90	14.49	13.04
Subject 5	11.11	20	4	94.2	83.33	15.29	12.74
Subject 6	x	46	4	308.7	92	9.72	8.94
Average	8.89				89.75	12.48	11.14

Table 5.2. Average Online Performance

For example, Subject 2 typed a total of 18 letters, 16 correct and 2 wrong. Then,

$$\text{Accuracy} = \frac{16}{18} \times 100 = 88.88\%$$

The total time for 18 decisions were 69.6 seconds. Then, typing rate in letters per minute is 15.52. This rate is achieved with 88.88% accuracy, due to the fact that 2 of the decisions were wrong.

The second accuracy versus rate calculation makes sure the subject types the exact letter he/she wants, so time spent on wrong classification

results are taken into calculation as lost time. This means, the subject has to correct the error, either by a backspace, or by retyping the intended letter. For example, if the subject intends to spell 'ABCD' and spells 'ABCE' instead, he must correct himself by either spelling a backspace and the correct letter (which in this case the number of total spelled letters would be 7, with 5 correct letters (A, B, C, D, and one backspace) and one error (E)), or where backspace is unavailable, retyping the correct letter. Then the resulting string would be 'ABCED', with 4 correct letters and 1 wrong letter. Note that the system is still prone to errors, but making sure that the subject writes all the intended letters has a meaning. Now, Subject 2 intended to type 16 letters, but instead typed 16 correct and 2 wrong letters, achieving his/her goal in 18 letters. The total time to type all the intended 16 letters were 69.6 seconds. Then, the typing rate for intended letters is 13.79 letters per minute.

On average, our subjects had a typing rate of 12.48 letters per minute with an accuracy of 89.75%.

Figure 5.19 shows the first type of online analysis performance of Subject 5 with classical matrix and Figure 5.20 shows the first type of online performance of Subject 5 with color matrix. Figure 5.21 shows the first type of online analysis performance of Subject 3 with color matrix. Figure 5.22 shows the first type of online analysis performance of Subject 4 with color matrix.

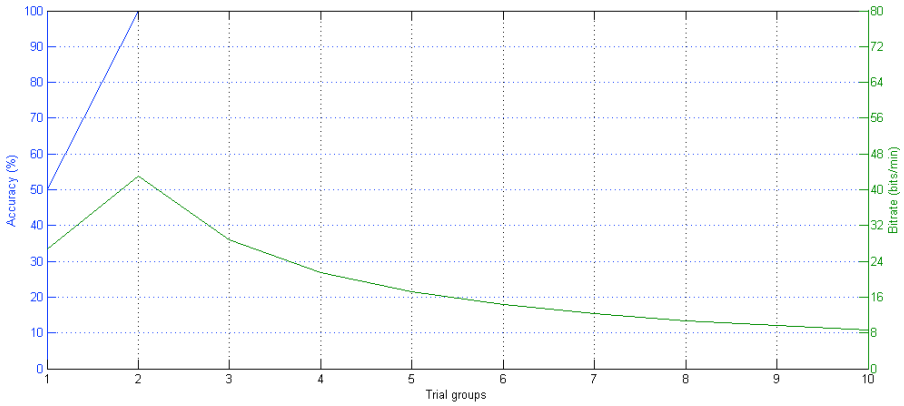


Figure 5.19. Online performance of Subject 5 with classical matrix

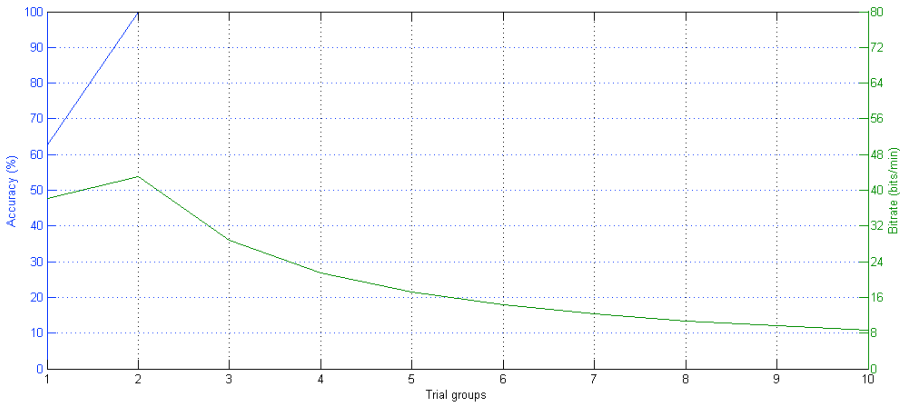


Figure 5.20. Online performance of Subject 5 with color matrix

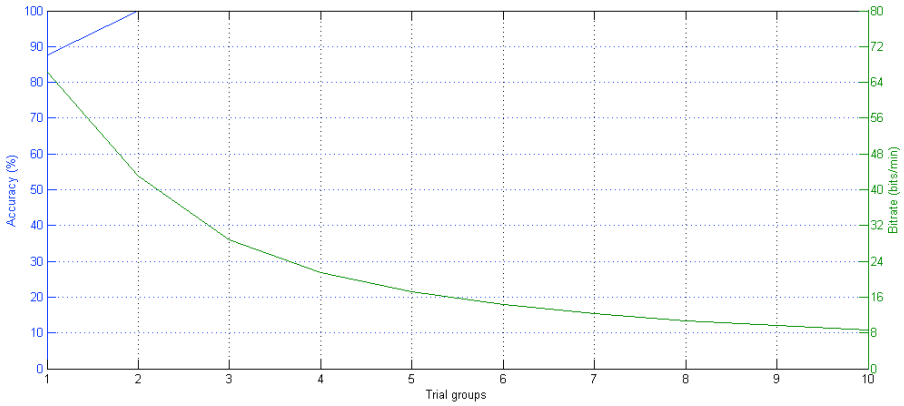


Figure 5.21. Online performance of Subject 3 with color matrix

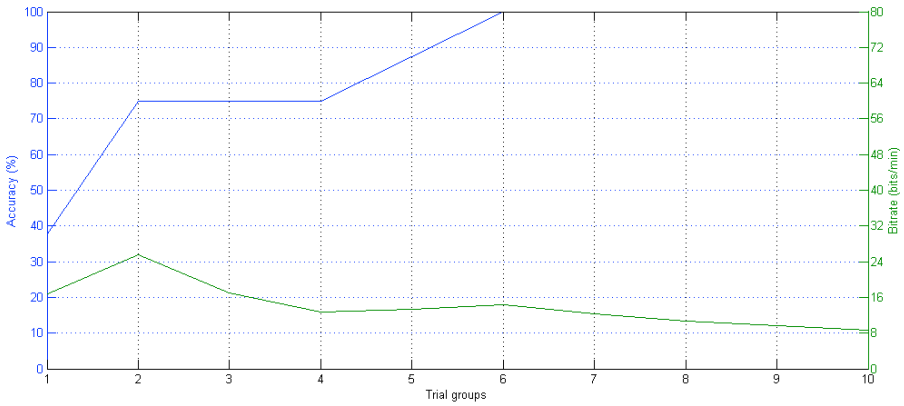


Figure 5.22. Online performance of Subject 4 with color matrix

5.4.2 Discussion

We have stated disadvantages of in-epoch data processing, in online analysis where the system is real-time, and then reported a performance

drop in most of the subjects. Then we showed online analysis results where color matrices were used as stimulus. The average speed is 12.48 letters/min for error ignorant results with accuracy of 89.75%, and 11.14 letters/min when errors are taken into consideration but the subject went on with typing until he typed all the letters he had intended first. The average offline performance of these subjects were 8.89 letters per minute for 100% accuracy. These results show that although there is a disadvantage in online analysis due to real-time processing, this disadvantage is compensated with our greedy algorithm that can produce fast results. Therefore, we see an increase in typing rate in online analysis. For online analysis, we had the opportunity to test color matrix performance versus classical matrix performance only with subject 5, and we see that there is a slight performance gain with color matrix.

We should note that the performance of our online algorithm is not at its best, as margins were fixed for all subjects. We believe that once a learning algorithm is developed for detecting optimum margins for subjects based on their training data, the performance would be much better. In other words, performance could be improves further by subject-adaptive selection of margins.

CHAPTER 6

CONCLUSION

6.1 Summary

In this thesis, we have laid out basics of EEG applications and signal processing in EEG context. We have presented a new end-to-end brain-computer interface, the P300 speller and evaluated its performance with both offline and online experiments. Compared to the published work so far, our results were superior in both cases. We have shown that, the P300 speller is a good means of communication for disabled subjects.

Tackling the practical problems of such a system, we have explored various choices and improvements in both signal processing and the stimulus side. Effects of different data filtering schemes and different modes of stimulation were investigated and results were reported.

6.2 Future work

6.2.1 Stimulus

Although we have presented our work in two schemes, one being the classical 6x6 matrix of gray/white letters and the other featuring random color flashes, there is a lot to be done in terms of stimulus experiments. Knowing there is little published work on stimulus alternatives, when designing our P300 stimulus software, we have kept in mind such opportunities, so our software allows many different customizations such as stimulus timing, matrix size and dimensions, cell contents, colors, flashing modes, etc. We are planning to do more experiments with different flashing modes, and especially, the round matrix.

6.2.2 Feedback

Another underdeveloped area of research in P300 speller is the means of feedback to the user. Most commonly, researchers wait for a predetermined time before displaying the result to the user, but we have shown in our work that this approach is redundant.

We have implemented the software infrastructure that allows us to display partial results as feedback, as the data is processed in real-time, as we think, this might increase the efficiency of the system. We have

experimented with such partial result feedback stimuli, but have not achieved a better performance. This is a field where a wide variety of improvements are possible.

6.2.3 Signal Processing

This area is where most of the effort in P300 speller research goes, and published work shows that there is no decisive answer for the question of which processing schemes to apply to subjects, as brain patterns and regulations of subjects differ, and their performance vary across different processing schemes.

In our work, we have chosen a fairly rugged, stable, good performance classifier, a Linear Discriminant Analysis with a Bayesian approach. This classifier features posterior density probabilities whose regularization parameters are learnt automatically from the training data. The research might be extended; more classifiers shall be tested and a study of their comparative performance shall be performed.

Furthermore, the performance of the decision-making algorithm we propose might be improved via application of machine learning techniques that deduce the necessary parameters and weights from the training data, therefore making the algorithm subject-specific.

Another area of interest is the difference between row and column classification results. Our work shows that rows are harder to classify then columns. Although we have implemented one classifier for all stimuli, it is sensible to incorporate two classifiers; one for columns and one for rows. In fact, a study in this context presented promising results for such an approach in [14].

Another important aspect is the definition and characterization of errors in classification. Studying on the wrong answers in our experiments, we saw that often, a letter adjacent to the intended target is selected as the answer. Furthermore, an important portion of errors included letters irrelevant to the target, positioned in a far corner of the matrix. The effects that cause this type of wrong classification should be investigated in more detail. We believe that, training a second classifier based on these errors should help in error-reduction or preventing erroneous feedback to the user.

Overall, the P300 speller is still an exciting field of research that has countless unanswered questions. We are planning to address some of the issues addressed here in more detail in the near future.

Bibliography

[1] *User Reference:P3SpellerTask*. Available: http://www.bci2000.org/wiki/index.php/User_Reference:P3SpellerTask

[2] A. Bashashati*, et al.*, "A survey of signal processing algorithms in brain-computer interfaces based on electrical brain signals," *J Neural Eng,* vol. 4, pp. R32-57, Jun 2007.

[3] J. D. Bayliss and D. H. Ballard, "A virtual reality testbed for brain-computer interface research," *IEEE Trans Rehabil Eng,* vol. 8, pp. 188-90, Jun 2000.

[4] J. D. Bayliss*, et al.*, "Changing the P300 brain computer interface," *Cyberpsychol Behav,* vol. 7, pp. 694-704, Dec 2004.

[5] H. Berger, "Uber das elektrenkephalogramm des menschen," *Arch Psychiatr Nervenkr,* vol. 87, pp. 527-570, 1929.

[6] BioSemi, "64 channels Surgical Medium/Small (Red/Yellow) 10/20 layout," in *http://www.biosemi.com/pics/cap_64_layout_medium.jpg*, ed.

[7] BioSemi, "64 channels Surgical Medium/Small (Red/Yellow) 10/20 layout," in *http://biosemi.com/pics/cap_surg645M_front_large.jpg*, ed.

[8] N. Birbaumer*, et al.*, "A spelling device for the paralysed," *Nature,* vol. 398, pp. 297-8, Mar 25 1999.

[9] C. M. Bishop and N. M. Nasrabadi, "Pattern Recognition and Machine Learning," *Journal of Electronic Imaging,* vol. 16, pp. 049901-2, 2007.

[10] T. P. Centeno and N. D. Lawrence, "Optimising Kernel Parameters and Regularisation Coefficients for Non-linear Discriminant Analysis," *J. Mach. Learn. Res.,* vol. 7, pp. 455-491, 2006.

[11] E. Donchin and D. B. D. Smith, "The Contingent Negative Variation and the Late Positive Wave of the Average Evoked Potential," *Electroencephalogr Clin Neurophysiol,* vol. 29, pp. 201-203, 1970.

[12] E. Donchin*, et al.*, "The mental prosthesis: assessing the speed of a P300-based brain-computer interface," *IEEE Trans Rehabil Eng,* vol. 8, pp. 174-9, Jun 2000.

[13] C. C. Duncan-Johnson and E. Donchin, "On quantifying surprise: the variation of event-related potentials with subjective probability," *Psychophysiology,* vol. 14, pp. 456-67, Sep 1977.

[14] H. B. Erdogan, "A Design and Implementation of P300 Based Brain-Computer Interface," MSc., Graduate School of Natural and Applied Sciences, Middle East Technical University, Ankara, 2009.

[15] 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, pp. 510-23, Dec 1988.

[16] U. Hoffmann*, et al.*, "Application of the evidence framework to brain-computer interfaces," *Conf Proc IEEE Eng Med Biol Soc,* vol. 1, pp. 446-9, 2004.

[17] U. Hoffmann, *et al.*, "An efficient P300-based brain-computer interface for disabled subjects," *J Neurosci Methods,* vol. 167, pp. 115-25, Jan 15 2008.

[18] M. Kaper and H. Ritter, "Generalizing to new subjects in brain-computer interfacing," *Conf Proc IEEE Eng Med Biol Soc,* vol. 6, pp. 4363-6, 2004.

[19] S. Kinoshita, *et al.*, "Long-term patterns of change in ERPs across repeated measurements," *Physiol Behav,* vol. 60, pp. 1087-92, Oct 1996.

[20] D. J. Krusienski, *et al.*, "Toward enhanced P300 speller performance," *J Neurosci Methods,* vol. 167, pp. 15-21, Jan 15 2008.

[21] A. Kübler and K.-R. Müller, "An Introduction to Brain-Computer Interfacing," in *Toward Brain-Computer Interfacing*, G. Dornhege, *et al.*, Eds., ed: MIT Press, 2007, pp. 1-25.

[22] D. J. C. MacKay, "Bayesian Interpolation," *Neural Computation,* vol. 4, pp. 415-447, 1992.

[23] S. Makeig, *et al.*, "Blind separation of auditory event-related brain responses into independent components," *Proc Natl Acad Sci U S A,* vol. 94, pp. 10979-84, Sep 30 1997.

[24] P. Meinicke, *et al.*, "Improving Transfer Rates in Brain Computer Interfacing: a Case Study," *NIPS,* pp. 1107-1114, 2002.

[25] G. R. Muller-Putz, *et al.*, "EEG-based neuroprosthesis control: a step towards clinical practice," *Neurosci Lett,* vol. 382, pp. 169-74, Jul 1-8 2005.

[26] E. Niedermeyer and F. H. Lopes da Silva, "EEG Recording and Operation of the Apparatus," in *Electroencephalography : basic principles, clinical applications, and related fields*, 5th ed Philadelphia ; London: Lippincott Williams & Wilkins, 2005, pp. 140-141.

[27] B. Obermaier, *et al.*, ""Virtual keyboard" controlled by spontaneous EEG activity," *IEEE Trans Neural Syst Rehabil Eng,* vol. 11, pp. 422-6, Dec 2003.

[28] D. Ravden and J. Polich, "Habituation of P300 from visual stimuli," *Int J Psychophysiol,* vol. 30, pp. 359-65, Nov 1998.

[29] G. Santhanam, *et al.*, "A high-performance brain-computer interface," *Nature,* vol. 442, pp. 195-8, Jul 13 2006.

[30] G. Schalk, *et al.*, "BCI2000: a general-purpose brain-computer interface (BCI) system," *Biomedical Engineering, IEEE Transactions on,* vol. 51, pp. 1034-1043, 2004.

[31] E. W. Sellers and E. Donchin, "A P300-based brain-computer interface: initial tests by ALS patients," *Clin Neurophysiol,* vol. 117, pp. 538-48, Mar 2006.

[32] E. W. Sellers, *et al.*, "A P300 event-related potential brain-computer interface (BCI): the effects of matrix size and inter stimulus interval on performance," *Biol Psychol,* vol. 73, pp. 242-52, Oct 2006.

[33] H. Serby, *et al.*, "An improved P300-based brain-computer interface," *IEEE Trans Neural Syst Rehabil Eng,* vol. 13, pp. 89-98, Mar 2005.

[34] C. E. Shannon and W. Weaver, *The mathematical theory of communication.* Urbana,: University of Illinois Press, 1964.

[35] S. Sutton, *et al.*, "Evoked-potential correlates of stimulus uncertainty," *Science,* vol. 150, pp. 1187-8, Nov 26 1965.

[36] M. Thulasidas and C. Guan, "Optimization of BCI Speller Based on P300 Potential," *Conf Proc IEEE Eng Med Biol Soc,* vol. 5, pp. 5396-9, 2005.

[37] e. a. Van Gestel, "Bayesian Framework for Least-Squares Support Vector Machine Classifiers, Gaussian Processes, and Kernel Fisher Discriminant Analysis," *Neural Computation,* vol. 14, pp. 1115-1147, 2002.

[38] W. G. Walter, *et al.*, "Contingent Negative Variation: An Electric Sign of Sensorimotor Association and Expectancy in the Human Brain," *Nature,* vol. 203, pp. 380-4, Jul 25 1964.

[39] C. Wang, *et al.*, "P300 brain-computer interface design for communication and control applications," *Conf Proc IEEE Eng Med Biol Soc,* vol. 5, pp. 5400-3, 2005.

[40] R. W. Williams and K. Herrup, "The control of neuron number," *Annu Rev Neurosci,* vol. 11, pp. 423-53, 1988.

[41] J. R. Wolpaw, *et al.*, "Brain-computer interfaces for communication and control," *Clin Neurophysiol,* vol. 113, pp. 767-91, Jun 2002.

[42] J. R. Wolpaw and D. J. McFarland, "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans," *Proc Natl Acad Sci U S A,* vol. 101, pp. 17849-54, Dec 21 2004.