Universidad Zaragoza

1542

Trabajo Fin de Grado

Adaptive Filtering Techniques for the Detection of User – Independent Single Trial ERPs in Brain Computer Interfaces

Autor/es

Cristina Leza Lahuerta

Director/es

Juan Pablo Martínez Cortés

Escuela de Ingeniería y Arquitectura (EINA) - Zaragoza

2016

Resumen

Brain Computer Interface (BCI) es el término inglés dado a una tecnología basada en el uso de señales cerebrales y su posterior procesado por diversos sistemas. Permiten, de esta forma, comunicación directa entre nuestro pensamiento y el mundo exterior. Innumerables aplicaciones pueden derivarse de este tipo de sistemas, siendo especialmente importantes en el ámbito de la biomedicina. Pacientes de severas enfermedades como esclerosis lateral amiotrófica (ELA), accidentes cerebrovasculares, parálisis cerebral o lesiones de la médula espinal, pueden beneficiarse de los sistemas BCI y recuperar la comunicación con su entorno.

Estos sistemas son altamente exigentes, teniendo principalmente tres problemas fundamentales en su desarrollo: rapidez, precisión y adaptabilidad.
Con el objetivo de conseguir una comunicación lo más realista posible, esta tecnología ha de ejecutarse rápidamente. Estos sistemas suelen precisar un tiempo considerable tanto de entrenamiento como de prueba para validar su funcionamiento. Además, se requiere una alta precisión para asegurar una correcta transimisión de información. Finalmente, dada la complejidad de estos sistemas, una vez implementados es conveniente que puedan adaptarse a distintos usuarios.

El objetivo de este trabajo será implementar un sistema BCI adaptativo que minimice o incluso elimine el tiempo de setup previo requerido por esta tecnología. Especificamente, el proyecto se centra en la detección de potenciales cerebrales relacionados con estímulos externos presentados en una única fase, donde una rápida adaptación a cada usuario es crucial para asegurar el éxito del sistema.

Existen diversos tipos de tecnología BCI dependiendo del paradigma usado. En este contexto, paradigma se requiere a cómo se va a estimular el cerebro para obtener señales de las cuales extraer la voluntad del usuario.
En este proyecto se desarrollará un sistema BCI basado en el potencial evocado P300. Esta es una onda cerebral, registrada mediante electroencefalografía, que presenta un voltaje positivo aproximadamente 300 ms después de que el estímulo sea presentado al usuario.
Este método usa dos tipos de estímulo. Uno de ellos tiene una alta probabilidad de ocurrencia mientras que el otro consituye un evento poco probable. Concretamente, esta respuesta es desencadenada entorno a la región parietal por el evento poco probable, que está relacionado de alguna forma con la voluntad del usuario.
Localizando dicha onda cerebral, el sistema será capaz de reconocer qué acción quiere

tomar quien esté usando la tecnología.

Un sistema BCI tiene varias partes fundamentales: adquisición de señal, preprocesado de señal, extracción de características, clasificación, interfaz y aplicación concreta del sistema.
Entre los diversos métodos existentes para adquirir señales cerebrales, el más utilizado es la electroencefalografía. La actividad eléctrica del cerebro puede ser medida y grabada. Mediante esta técnica, dicha actividad es medida colocando electrodos a lo largo del cuero cabelludo. Cada uno de los electrodos posicionados constituirá un canal de entrada de datos. Las oscilaciones grabadas de los potenciales eléctricos del cerebro son conocidas como electroencefalograma (EEG).
Siguiendo en el sistema, una parte crucial es el preprocesado de señal. Las señales EEG presentan numerosos artefactos y baja relación señal a ruido (SNR). Los pasos de preprocesado son entonces esenciales para eliminar o minimizar al máximo posible los artefactos y aumentar la SNR.
Existen numerosos tipos de sistemas BCI. En función de dicho tipo, varias características pueden extraerse de una misma señal cerebral para identificar patrones y separar los datos en diferentes clases. La elección de las características a extraer es especialmente importante, tanto por su clase como por el número de características, para el sistema ya que una decisión incorrecta podría llevar incluso a empeorar el desarrollo del sistema BCI.
El siguiente paso es clasificar las características extraídas para obtener las intenciones del usuario del sistema. Los principales problemas de los clasificadores son dos: dimensionalidad (en inglés, curse of dimensionality) y compromiso entre sesgo y varianza. El problema de la dimensionalidad se refiere a que la cantidad de datos requeridos para obtener un rendimiento dado incrementa proporcionalmente con el número de características usadas para caracterizar la señal cerebral.
Finalmente, la interfaz dependerá de la aplicación dada al sistema y es el medio de interacción directa con el usuario.

Gran parte de este trabajo está centrada en el preprocesado de la señal cerebral obtenida. Como se ha mencionado previamente, la capacidad de adaptación de un sistema BCI es crucial para el desempeño del mismo. Así, se desarrollarán varias técnicas para obtener una señal cerebral lo más limpia posible y en la cual los potenciales buscados estén resaltados.
En primer lugar, se realizará un filtrado de la señal para mantener los componentes frecuenciales donde se encuetran las señales de interés y reducir en la medida de lo posible el ruido presente en la señal.
Después, varias técnicas cuyo objetivo principal es reducir la dimensionalidad de los datos serán aplicadas a la señal filtrada.
El primer conjunto de métodos emplea análisis estadísticos para separar los datos o bien en sus componentes principales y de mayor importancia, Análisis de Componentes Principales (ACP, en inglés, PCA), o en componentes independientes entre sí, Análisis de Componentes Independientes (ACI, en inglés, ICA).
El siguiente grupo de técnicas evaluadas consisten en filtros espaciales que, además de reducir la dimensionalidad de los datos, incrementan la SNR mediante una com-

binación lineal de los diferentes canales de entrada de datos originales. Los dos procedimientos implementados en este proyecto son el filtro xDAWN y su mejora adaptativa, el filtro axDAWN.
La idea básica tras ambas versiones es realzar la respuesta síncrona al estímulo correspondiente al objetivo del usuario. Este proceso se divide en dos pasos fundamentales: primero se estiman las respuestas síncronas y después se diseñan los filtros que realzan dichos potenciales.
El filtro axDAWN mejora el diseño de su predecesor incluyendo una adaptación de los coeficientes de los filtros usados mediante el algoritmo Recursive Least Squares (RLS).

Siguiendo el curso del sistema BCI, el siguiente paso es la extracción de características de la señal cerebral ya preprocesada.
El principal método usado y en el cual se basa todo el sistema BCI es la transformada Wavelet (en inglés, WT). Dicha transformada se diferencia de la clásica transformada de Fourier en que esta es local tanto en tiempo como en frequencia. Esta característica de la transformada la hace apropiada para el análisis de señales cerebrales, que requieren una alta resolución tanto en tiempo como en frecuencia.
Concretamente, la WT será aplicada en un análisis en varias resoluciones conocido, en inglés, como Multi Resolution Analysis (MRA). La gran ventaja de esta técnica es la posibilidad de separar la señal en diferentes resoluciones frecuenciales. Esto es de gran importancia cuando la señal a analizar es un EEG ya que distintas actividades cerebrales se corresponden con diferentes bandas frecuenciales. En concreto, el potencial P300, en el cual se basa el sistema BCI desarrollado en el trabajo, tiene sus componentes más prominentes localizadas entre 0 y 4 Hz.
Una vez extraída la banda de interés para el sistema, se requiere una herramienta capaz de resaltar la respuesta síncrona al estímulo buscado.
La entropía es una medida de la incertidumbre de una fuente de información introducida por Shannon. Dada esta interpretación de la entropía, esta puede ser considerada también una medida de desorden. Cuanto más caótica es una distribución, mayor es su entropía.
Aplicado al campo de BCI, es de esperar que la entropía de una señal cerebral tenga valores altos ya que esta no tiene una distribución estadística establecida y puede considerarse aleatoria.
Sin embargo, el potencial P300 puede verse como una transición de un estado desordenado a uno ordenado en la señal cerebral dado que las oscilaciones correspondientes a dicho potencial están sincronizadas. De esta forma, la presencia de dicho potencial podrá ser advertida en un decremento del valor de la entropía de la señal.
A partir de la definición de Shannon, diversas variantes de la entropía han sido descritas. Concretamente, una variación que se ajusta mejor al sistema, dada la distribución estadística de la aparición de estímulos en la señal cerebral, es la introducida por Renyi. Esta nueva definición de entropía será la usada en este trabajo.

Dentro del gran repertorio de clasificadores usados en los sistemas BCI, en este trabajo el algoritmo elegido será una máquina de vectores de soporte (en inglés, Support Vector Machine, SVM). Una SVM será entrenada con parte de los datos obtenidos para crear un modelo que sea capaz de predecir a qué clase pertenece cada una de las muestras siguientes.

Finalmente, respecto a la interfaz y la aplicación concreta del sistema, en este trabajo se desarrollará un deletreador.
La interfaz constará de una matriz 6x6 conteniendo tanto números como letras. La finalidad de dicho sistema es que este sea capaz de detectar en qué letra o número el usuario se está concentrando. De esta forma, cualquier persona con dificultades en la comunicación será capaz de transmitir cualquier palabra o número simplemente fijando su pensamiento en cada uno de los caracteres que lo formen.
Cada una de las filas y columnas de la matriz se iluminará de forma aleatoria, dejando un intervalo entre cada una de estas iteraciones (en inglés, Inter Stimulus Interval, ISI). El intervalo ISI puede tener un gran impacto en el desarrollo del sistema, y será estudiado en varios ensayos con diferentes valores ISI.
Entre una distribución tan aleatoria, el resalte de la fila o columna donde está contenido el caracter en el que el usuario está pensando se corresponderá con un evento poco probable. Así, esta ocurrencia será precisamente el tipo de estímulo que es capaz de desencadenar el potencial P300.
Gracias a las técnicas brevemente descritas anteriormente, el sistema podrá encontrar dicha respuesta síncrona al estímulo y, en consecuencia, obtener el caracter en el cual el usuario estaba centrando su atención.
Además de la interfaz, el sistema requiere un montaje experimental para poder grabar las señales cerebrales a analizar. Dicho montaje se trata de un método no invasivo, formado por 16 electrodos colocados a lo largo del cuero cabelludo en posiciones específicas dictadas por el estándar internacional llamado, en inglés, 10-20 system.
Concretamente y dentro de todas las posiciones descritas en dicho estándar, los electrodos estarán colocados principalmente en el lóbulo parietal, donde el potencial P300 puede ser encontrado.
Entre dichos electrodos y el equipo encargado de analizar el EEG grabado, un amplificador tratará con la típica baja amplitud de las señales cerebrales.

Con el objetivo de evaluar todas las técnicas, tanto de preprocesado como de análisis, y la interfaz propuesta, dos conjuntos de datos serán usados.
El primero de ellos se corresponde con señales previamente grabadas en un sistema igual al descrito anteriormente, y pertenecientes al proyecto Brain Neural Computer Interaction Horizon 2020 (BNCI Horizon 2020).
Además, un experimento involucrando al sistema específicamente desarrollado para este trabajo se llevará a cabo con 10 voluntarios.
Este experimento consta de tres fases en las que cada uno de los voluntarios usará el deletreador para dictar palabras predefinidas, de 5 letras cada una.
La primera fase presentará al usuario únicamente dos palabras con el objetivo de asegurar que este ha entendido correctamente el funcionamiento del sistema.
Después, en la siguiente etapa, el voluntario deletreará tres palabras.
Finalmente y buscando evaluar el impacto del ISI en el sistema, la tercera y última fase volverá a constar únicamente de dos palabras, esta vez con un intervalo entre

resaltes de filas y columnas más amplio que en las dos pruebas anteriores.

Para concluir el trabajo, se presentarán los resultados obtenidos para ambos conjuntos de datos.
En primer lugar, las suposiciones teóricas acerca de la relación entre la entropía de las señales cerebrales y la presencia de estímulo será validada. La descomposión MRA y el cálculo de la entropía, sin ningún tipo de preprocesado, es capaz de mostrar la presencia de estímulos en las señales grabadas. Así, las técnicas de extracción de características descritas prueban ser válidas para un sistema BCI basado en el potencial P300.
Posteriormente, el efecto de los diferentes métodos de preprocesado será mostrado. La ya evidente relación teórica previamente descrita se verá realzada gracias a los métodos implementados. Gracias al preprocesado, el desempeño del sistema será ampliamente mejorado.
Vistos estos primeros resultados, el sistema será finalmente evaluado por medio de porcentajes de precisión de acierto del caracter pensado por el usuario.
Dichos porcentajes probarán que el rendimiento del sistema es prácticamente perfecto para todos los casos evaluados.

El trabajo desarrollado es una presentación de los sistemas BCI e involucra el entendimiento tanto de su funcionamiento como de las señales cerebrales en las que este está basado.
Para este proyecto, cada una de las partes que forman un sistema BCI serán desarrolladas.
Se estudiarán diversas técnicas de procesado de señal para ser aplicadas en varias partes del sistema.
Además, una interfaz adecuada será creada para validar los métodos desarrollados y evaluar el desempeño del sistema.
El resultado final del trabajo será un sistema BCI completo con una alta tasa de acierto y, por tanto, con gran potencial de aplicabilidad.

TECHNICAL UNIVERSITY OF DENMARK

Bachelor Thesis

Adaptive Filtering Techniques for the Detection of User - Independent Single Trial ERPs in Brain Computer Interfaces

Supervisor
Sadasivan Puthusserypady

Author
Cristina Leza Lahuerta

June 2016

Adaptive Filtering Techniques for the Detection of User - Independent Single Trial ERPs in Brain Computer Interfaces

This report was prepared by:
Cristina Leza Lahuerta, s150802

Supervisor:
Sadasivan Puthusserypady, Associate Professor, Ph.D.

Department of Electrical Engineering
Technical University of Denmark
2800 Kgs. Lyngby
Denmark

Project period:	February 1st 2016- June 17th 2016
ECTS:	15
Education:	Bachelor of Science
Field:	Biomedical Engineering
Class:	Open Source
Copyrights:	©Cristina Leza Lahuerta, 2016

Signature

[illegible]

Cristina Leza Lahuerta

Acknowledgments

In relation to this thesis, I would like to thank a number of people.
First of all, I would like to thank my supervisor, Sadasivan Puthusserypady, for his help and guidance.
I would also like to thank all the fellow students who have helped me with the project.
Finally, a special thanks goes to my family and friends for their support during all the development of the thesis.
Without you, this would have not been possible.
Thank you.

Abstract

Brain Computer Interfaces (BCI) are systems that use directly brain signals to communicate with the external world. They are challenging systems in constant evolution. The main three problems to address in such systems are: speed, accuracy and adaptability.
The understanding of different types of BCIs, EEG signals and how these two are connected to each other is essential to design a perfect system.
P300 based BCI systems are known by its ease of implementation and use. However, adaptive techniques would significantly improve their performance. Studies are conducted on different existing techniques and their performance are evaluated. Also, new features for P300 based systems are researched.
A P300 speller was designed and an experiment conducted with 10 subjects was used to evaluate the system proposed.
All in all, a complete P300 based BCI system was the subject of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is a truth universally acknowledged, that communication is a key tool for every human being. Technological progress has led to a new and outstanding conception of communication. Nevertheless, nothing can compare to sharing thoughts and intentions with the brain itself.

Over the past years, a promising yet maybe unnoticed technology has been under development: Brain Computer Interface (BCI). Countless applications, ranging from medical purposes to leisure, can benefit from BCI. These systems read signals directly from the brain, allowing communication and control over one's surroundings. This ability has a radical impact on people suffering from severe neuromuscular disorders such as amyotrophic lateral sclerosis (ALS), brainstem stroke, cerebral palsy or spinal cord injury. BCI opens the door to new ways of therapy and even rehabilitation.

1.1 Objectives

How to get the most out of this technology? BCI systems usually require a significant amount of training and testing time since their calibration steps differs from one user to another. Huge improvements could be made if the system would adapt to the particular subject making use of it.

The purpose of this project is to develop an adaptive system so as to minimize, and even remove, this setup time. Specifically, the project will focus on single-trial detection of Event Related Potentials (ERPs) where fast adaptation to the user is essential.

This thesis also includes the development of a real BCI system. An experiment on this system will be conducted, acquiring offline data to test its validity.

Within the wide field of BCI, this project is focused on P300 based systems, characterized by their ease of implementation. Besides, they are known to require little or no training from the user so adaptive schemes could highly improve their applicability to users. Specifically, the tested system will be a P300 Speller.

All in all, the aim of this project is to improve BCIs by including adaptation for detection of user. The resulting systems will not only be faster, but also more accurate since all the detection process will be personalized to each subject.

1.2 Contributions

As it has been previously mentioned, two of the most important aspects to be fulfilled by a BCI system are: speed and minimal training and calibration. In this thesis, several techniques will be proposed to:

- Implement an adaptive BCI system with little or no calibration time requirements.
- Seek new features to characterize the P300 response.
- Improve the existing P300 based BCI system paradigms so as to make them both user-friendly and more suitable for computation.

1.3 Overview of the Thesis

The thesis is structured as follows:

- **Chapter 2**: the reader will have a general insight into the human brain, how it works and how BCI systems can extract information from it.
- **Chapter 3**: BCI systems will be introduced. An overview of the whole system is presented, followed by an introduction of the main paradigms used.
- **Chapter 4**: a literature review will present some of the main techniques currently being employed in BCI systems.
- **Chapter 5**: the methods developed for this thesis will be explained in detail.
- **Chapter 6**: presentation of the most commonly used tool for P300 based BCI systems. Following this introduction, the interface design for the project will be illustrated. Last, the conducted experiment will be explained.
- **Chapter 7**: the results acquired from the techniques developed in the thesis will be presented.
- **Chapter 8**: based on the previous chapter, a comparison between the different implemented methods and a discussion on the results will be provided in the eighth chapter.
- **Chapter 9**: summing up the conclusions obtained from research work carried out.

Chapter 2

Clinical Background

Human brain is the central point to every BCI system. To understand how such systems work, it is then required to have some basic knowledge about the background they work on. Accordingly, this chapter will first deal with a short description of the brain physiology, explaining its functionalities. Following this, a brief description of the EEG, main tools of BCI systems will be presented.

2.1 Brain Physiology

No organ can be compared to the brain. It controls all body functions and allows for communication with the external world, by receiving and transmitting information. The complexity of the brain is immense, containing more than 100 billion neurons which are highly interconnected. The brain can be divided into several parts, each of them having its own function: cerebrum, cerebellum and brainstem. The most refined functions of the brain are performed in the cerebrum, which is divided into two hemispheres connected by the Corpus Callosum.

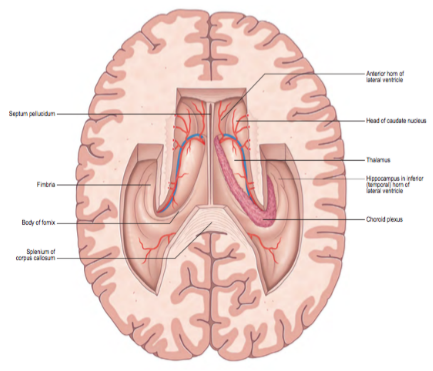

Anterior horn of lateral ventricle
Septum pellucidum
Head of caudate nucleus
Thalamus
Fimbria
Hippocampus in inferior (temporal) horn of lateral ventricle
Choroid plexus
Body of fornix
Splenium of corpus callosum

Figure 2.1: Horizontal section of the cerebrum (Gray's Anatomy [1])

The cerebral cortex is the outermost layer of the cerebrum controls most of the information processing. It can be divided in four major lobes, which can be depicted in figure 2.2, each of them dealing with specific functions [2]:

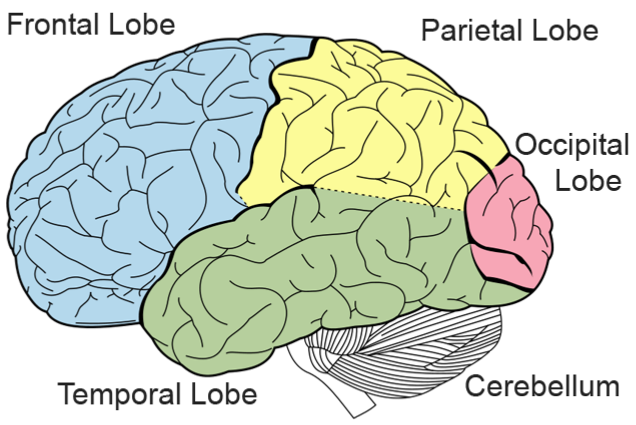


Figure 2.2: Brain Lobes (Gray's Anatomy [1])

1. The frontal lobe is responsible for behaviour, such as judgment and decision making.
2. The parietal lobe processes sensory input. Therefore, they manage the information coming from eyes, ears, tongue or skin; dealing with spatial sense and navigation.
3. The temporal lobe manages the sensory input and retains visual and auditory information as memory.
4. The occipital lobe represents the brain's visual processing center.

As it will be later seen, the BCI system developed in this project deals with visual stimuli. Therefore, the occipital region is the most important one for this project.

2.2 Electroencephalogram

The electrical activity of the brain can be measured and recorded. Electroencephalography is the neurophysiological technique that measures this activity by placing electrodes on the scalp. It is a widely used non-invasive method that is useful not only to monitor normal brain activity, but also to detect neurological disorders and assist brain activity studies such as sleep analysis. The recorded oscillations of such brain electric potentials are known as electroencephalogram (EEG), and they are caused by the synchronous electrical charge activity of a huge number of brain cells called neurons. Since this activity has to be synchronized, neurons need to have similar spatial orientation so as to create waves to be detected. Pyramidal neurons of the cortex are usually seen as the ones to produce the most prominent EEG signals since they are well-aligned and they fire-up together. The recorded signal is then the summation of the potentials provoked by these neurons. Using appropriate devices, these potentials can be measured, constituting the EEG signals.

EEG is usually recorded using the standard 10-20 system, where '10' and '20' refers to the distance between electrodes. The following figure illustrates the system.

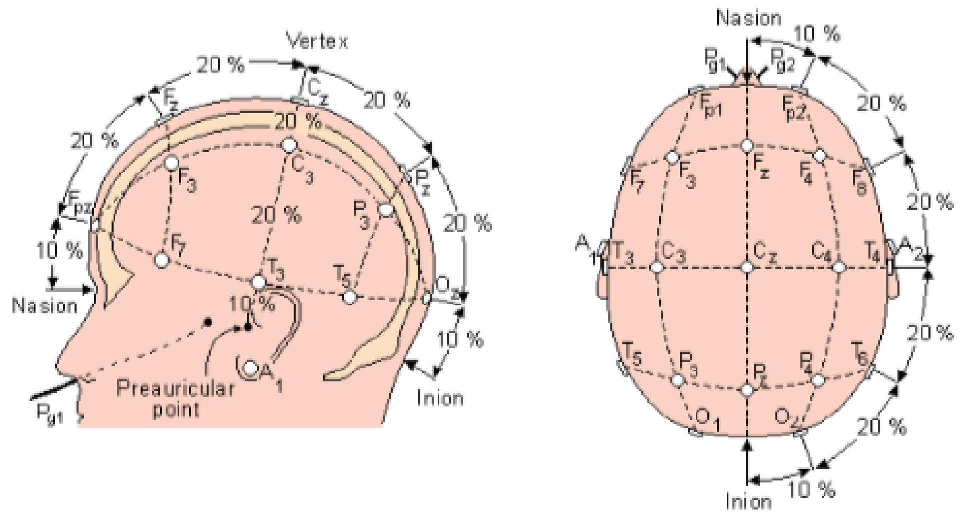


Figure 2.3: Electrode Placement System (Source: BCI org)

The first letters of the electrodes placement indicate the region of the scalp where the electrode is located: Fp - prefrontal, F - frontal, C - central, P - parietal, O - occipital and T - temporal. As it can be seen, C does not stand for any lobe, but it is rather used for identification. The midline is represented by the letter 'z' and the numbers accompanying the lobe, or central, positions represent electrodes on the right (even numbers) or the left (odd numbers) hemispheres.

After the EEG signal is recorded, typically at a rate of 256 Hz, it needs to be amplified due to the low voltage amplitude of brain signals (usually μV or mV)

Most BCIs systems use the EEG as the modality. However, there are BCI systems based on other modalities such as Magnetoencephalograhy (MEG), functional magnetic resonance imaging (fMRI) and near infrared spectroscopy (NIRS) as well. MEG measures small changes in the magnetic field associated with brain activity. fMRI measures the blood oxygenation level-dependent (BOLD) signals, which are also associated with cortical activation. Another hemodynamic technique is NIRS, based on the different oxygen levels of the blood. As mentioned, these techniques have also been used in BCIs. Nevertheless, they suffer from serious drawbacks that make using electrodes to measure EEG a better choice. MEG and fMRI are quite expensive techniques and also, both fMRI and NIRS have poor temporal resolution[3].

EEG signals can be classified in two main categories depending on how the brain signals are generated: Event-Related Potentials (ERPs) and Spontaneous EEG activity. ERPs or Evoked Potentials are stereotypical responses to external stimuli. These stimuli can be either sensory, cognitive or motor events. These evoked potentials are usually obtained by applying such stimulus to a subject, and can be later recorded and applied to a number of applications. Within the field of BCI, ERPs are of utmost importance and will be later described in more detail. Spontaneous brain activity, which can be triggered by any outside activity, in the absence of any specific task or stimuli.

EEG waves have characteristic frequency ranges, each of them corresponding to specific spatial distributions and associated with different brain functions.

The delta waves are associated with the lowest frequency of oscillation, 0-4 Hz. Delta waves are characterized by their high amplitude and are commonly associated with deep sleep.

The theta waves can be found in the 4-7.5 Hz range and it is related to light sleep or relaxation.

Alpha waves, found in the 7.5-12.5 Hz range, are associated with awake state but with little mental activity. Also, alpha waves can be found when a person closes his/her eyes. Mu waves, also within the 7.5-12.5 Hz range correspond to the synchronized patterns found in the part of the brain that controls voluntary movement. Beta waves, within 12.5-30 Hz, are related to full consciousness. Therefore, insufficient beta activity can be a sign of neurological disorders such as insomnia. Finally, gamma waves, also in the 25-100 Hz are associated with various types of learning processings in the brain[2].

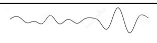
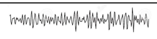
Type of Activity	Band	Typical Waveforms
δ	0 - 4 Hz	
θ	4 - 7.5 Hz	
α	7.5 - 12.5 Hz	
μ	7.5 - 12 Hz	
β	12.5 - 30 Hz	
γ	25 - 100 Hz	

Table 2.1: Brain waves in EEG

2.3 Conclusions

This chapter presented some basic concepts about the human physiology BCI systems are based on. The following chapter will introduce these systems, their components and how they operate.

Chapter 3

Brain Computer Interface

This chapter will begin with an introduction to a complete BCI system. The most commonly used paradigms used in BCI systems, being the one used for this project the P300 potential, will be also described briefly in this chapter.

3.1 Overview

BCI is a revolutionary technology that allows communication with the external world only with the help of our thoughts. This will not be possible without all the essential components forming a BCI system, depicted in the figure 3.1 below. The components of the system include signal acquistion, signal preprocessing, feature extraction, classification, application interface and application.

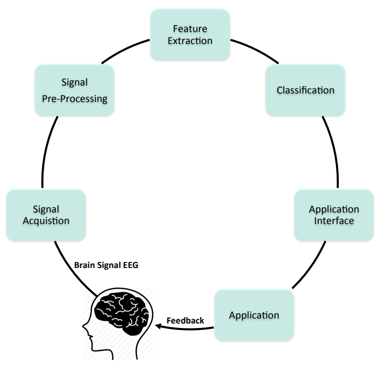


Figure 3.1: Overview of a BCI system

3.1.1 Signal Acquisition

This consists of the EEG electrodes, along with an adequate amplifier, are capable of digitizing and storing brain signals. As mentioned in previous chapters, there are several forms of brain signal acquisition. However, EEG electrodes are the most widely used technique. Specifically, real-life applications mostly use gel based electrodes since they provide more robust signal quality. Nevertheless, they have two main disadvantages: long montage time and need of washing both cap and head afterwards. An alternative to gel based electrodes is the use of dry electrodes. This technology, still under study, is inferior to the other one in the sense of obtained signal quality and user comfort. Recent studies have though proved that dry electrodes are acquiring more and more importance in the BCI field [4].

3.1.2 Signal Preprocessing

Raw EEG signals present two main problems to be solved: artifacts and low Signal to Noise Ratio (SNR). Processing steps are essential to remove/minimize the artifacts as well as to improve the SNR.

- Artifacts: EEG signals are often contaminated from electrical activity produced by non-brain sources. These signals can come from ocular movements (electrooculogram, EOG), electromyography (EMG) or even heartbeat (electrocardiogram, ECG). Among these sources, it can be considered that, due to its proximity to the scalp, ocular movements are the most significant artifacts of all the above mentioned.

- Low SNR: while recording EEG signals, there are a wide number of noisy sources that can affect the quality of the recorded signals. Besides, brain signals have significantly lower amplitude compared to the noise and so, the acquired SNR for raw EEG signals is not sufficient to process them correctly. The main source of noise the project will be dealing with is the power line interference (found at 50 or 60 Hz).

3.1.3 Feature Extraction

In order to achieve the best BCI performance, an appropriate choice of features needs to be performed. The features extracted from EEG signals allows to recognize specific patterns and separate the data into different classes. Not every feature is relevant for the task of classification and, in fact, an excessive number of features could worsen the performance of the system [5]. There are many techniques for feature extraction in BCI systems, some of the most important will be described in subsequent chapters.

3.1.4 Classification

Given a set of extracted features, they have to be later classified to detect the user's intentions. Classifiers can be divided into many categories. Generative-discriminative classifiers differentiate from each other because the former computes the likelihood between different classes while the latter discriminate classes. Classifiers can also be either static, if they do not have temporal evolution, or dynamic if they do. Finally, stable classifiers have low complexity and small variations in the training data set do not excessively affect their performance. On the contrary, unstable classifiers are highly dependent on changes on the training set.

The two main problems that affect classifiers are the curse of dimensionality and the bias-variance tradeoff. The curse of dimensionality refers to the increasing amount of data required to achieve a good classification performance when the dimensionality of the feature vectors increments. The bias of a classifier is the difference between the expected value of its output and the actual value of the parameter being estimated. The variance represents the sensitivity of the classificator to small changes in the training set. Stable classifiers tend to have high bias and low variance, and vice versa. EEG signals are known to be non-stationary. Therefore, it is preferred to choose a classification technique that yields the lowest variance as possible [5] The most widely used classification methods will be later explained, along with the method chosen for this project.

3.1.5 Application Interface

Once the EEG signals have been correctly classified, the application interface decides what to do next, depending on the kind of purpose the specific type of BCI has. In the case of offline BCI systems, this could be the last step of the procedure, when the results of the processing are evaluated.

3.1.6 Application

This last step is specially important for online BCIs since it is now when the results of the system can be assessed. For example, a BCI system controlling a wheelchair will now be able to move it in the direction of the user's desire thanks to all of the previous acquisition and processing of the EEG signals.

3.2 BCI Paradigms

This section describes briefly the most commonly used paradigms that are used in BCI systems.

3.2.1 Motor Imagery

Movement affects brain activity, but also the imagination of movement can elicit a response in the brain. Just by differently imagining the kinesthetic movement of somatic body parts (left or right hand, for example), discriminative brain patterns can be obtained. Sensorimotor rhythms (SMR) refer to oscillations in brain activity recorded from somatosensory areas. A specific event can either decrease or increase the SMR, being those events called Event-Related Desynchronization (ERD) and Event-Related Synchronization (ERS), respectively. These patterns can be produced solely by the imagination of movement. During those intervals of imagination, the EEG will show activity especially in the mu and beta bands.

The key to this paradigm is that the patterns provoked by imagination are similar in their topography and spectral behavior to those elicited by actual movements. One of the major drawbacks of motor imagery is that, since each somatic body part generates a different brain pattern, an effective classification algorithm is required [3].

3.2.2 Visual Evoked Potentials

Visual Evoked Potentials (VEP) are fixed delayed changes in the electrical potentials in the brain created by a visual stimulus (e.g. a flickering square). Therefore, these systems can identify a target the user is looking at. In such a system, each target needs to be identified with a unique sequence that will provoke a unique pattern in the brain signal [6].

Steady State Visual Evoked Potentials (SSVEP) are especially important within the field of VEP based BCIs. The main characteristic of such systems is that the stimulus consists of a repetitive visual stimulus. Typically, the frequency range of the stimulus varies from 3.5 to 75 Hz[7]. These stimuli will generate electrical activity at the frequency of stimuli and its harmonics. Therefore, SSVEP systems are quite easy to implement just by inspecting the power spectrum of the EEG.

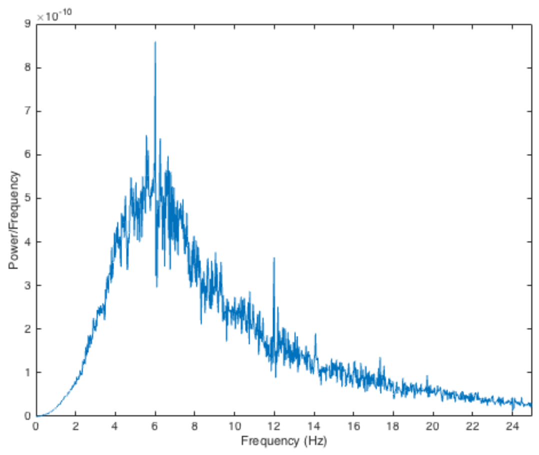


Figure 3.2: EEG Power Spectrum with SSVEP stimulus at 6 Hz

3.2.3 P300

The P300 response is an ERP response characterized by being a large positive wave appearing approximately 300 ms after the stimulus. Its amplitude is dependent on the attention required to process the given stimulus: the more focus needed, the higher the amplitude. The P300 wave has been traditionally elicited by the “oddball paradigm” [8].

This method presents the subject with two types of stimuli. One of them has a large probability of appearance and the other one is a low probability event. The P300 response will be triggered precisely by those low probability occurrences. Self-relevant targets are found to provoke higher amplitude responses, as well as explicitly instructed targets [9].

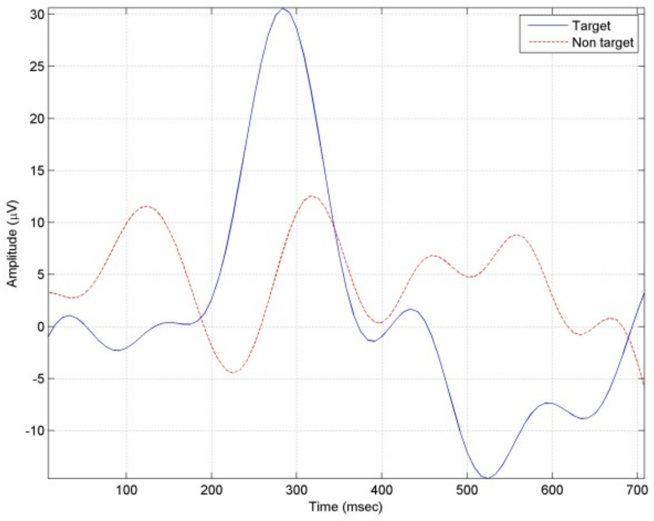


Figure 3.3: P3a and P3b responses

This last scenario is precisely the basis of the experiment conducted in this thesis to test the algorithms that have been developed. Behavioral tests have proved that this response can be found even 400-600 ms after the stimulus is presented [9]. The P300 latency also provides information about the duration of the stimulus since this wave shows the modifications in the neuronal activity taking place during the cognitive process [10]. This latency will also be involved in the amplitude of the response, as it has been found that higher inter stimulus interval yields larger P300 wave amplitudes [11].

As for the scalp distribution of the P300 wave, it is often described as the amplitude change over the midline electrodes, increasing from the frontal to the parietal lobe [11].

The P300 wave is known to have two subcomponents: P3a and P3b. The first potential, P3a has been commonly considered to be related to frontal lobe activation. The initial target requiring frontal attention then elicits this P3a wave. Subsequently, the P3b potential is presented when the attentional resources are allocated for memory updating in association cortex. Therefore, the P3b response is correlated to the temporal and parietal lobes[12].

This project will be focusing only in the P300 wave and not in its subcomponents.

Chapter 4

State of the Art

A BCI is an alternative pathway for an individual's thoughts and intentions. Therefore, its functionality is completely based on EEG signals. The system process goes from signal acquisition to their translation into appropriate commands. However, this last step cannot be reached without adequate feature extraction and later classification of the analyzed patterns.

In the specific scenario of adaptive BCI systems, several combination of signal processing and classification methods have been researched. Some of the newest and most innovative will be described in this section, as well as other non-adaptive but effective methods.

This chapter will also cover a study on the different implementation of P300 based BCI systems and the impact different design parameters have on their performance.

4.1 Review on Signal Processing

Sykacek et *al.* [13] propose the use of variational Kalman filtering as a technique to perform and also adaptive Bayesian classifier. The non-stationary nature of EEG signal represents a problem to every BCI systems. Their algorithm is based on the probabilities of different cognitive states present in the EEG to adapt the classifier. An adaptive classifier responding to changes in the brain activity statistically improved the performance of BCI systems. However, the results are unclear about the cognitive tasks performed during the experiments.

Kim and Jo [14] developed a real-time motion artifact detection for ambulatory BCI. Their approach is also based on adaptive Kalman filtering. Nevertheless, the filters are adapted to remove motion-related noise. The results showed that the adaptive filtering was able to improve the accuracy of BCI systems. However, this approach is designed for wireless BCI applications and, consequently, it does not have much relevance in a static P300 speller.

Tu et *al.* [15] present a combination of Common Spatial Filter (CSP) as a spatial filter and a wavelet filtering to enhance the SNR of single trial VEPs. The resulting system was significantly improved for the same subject but it still needed further amelioration in between-subjects trials.

Sun and Zhang [16] show a full subject-adaptive BCI system. In their system, a subject-oriented training procedure is implemented before the actual controlling of the system begins. The use of preprocessing techniques such as FastICA yields good results both in offline and online analysis. Nonetheless, a training setup is still necessary.

Turnip et *al.* [17] performed an real-time feature extraction for P300 waves based on adaptive non-linear filtering: adaptive non-linear principal component analysis (ANPCA). The experiment also tried different ISI to test the performance of their method. The results showed that the mixed sources were succesfully separated and an ISI value of 350 ms provided the most accurate results.

Bougrain et *al.* [18] conducted a study on the appropriate band pass filter to be used in the preprocessing step. According to their study, a band pass filter with upper cutoff frequency of 15 Hz improved the performance of the system. This represents a compromise between removing as much noise as possible without losing important information about the signal.

Woehrle et *al.* [19] designed an adaptive spatial filter, specially suited for ERPs like the P300 wave. They base their algorithm on a previously described spatial filter, xDAWN [20]. Then, they include the adaptivity by means of Recursive Least Squares (RLS) updates of the filters coefficients. This algorithm will be explained in detail in following chapters. The experiment performed to assess this method was a labyrinth game and the users were trained in the task. The subjects wore a head mounted display (HMD), displaying the labyrinth as well as stimuli symbols for the oddball task. The Inter Stimulus Interval (ISI) was significantly high (1000 ms). The results showed that the system improved with the inclusion of adaptation in the xDAWN filter. The conclusion of the study was that this regularized version was specially beneficial in the initialization of the algorithm when few data is available.

Krell et *al.* [21] further improved the two previous spatial filtering techniques. Their technique is a regularized version of aXDAWN (raXDWAN). Other filtering techniques, such as CSP, use regularization techniques to further remove noise and avoid overfitting. rAXDAWN modification over the previous system is based only in the initialization of the system. Therefore, it is quite easy to implement and could be relevant in this project. This study showed the great importance of the initialization parameter in this family of filters.

4.2 Review on the Interface

Regarding the interface used in the project, P300 Speller, several studies have been looking into the impact the ISI has on the performance of the system. Farwell and Donchin [22] claim that systems with higher ISI are more efficient. In their study, they find the P300 response based on peak detection. Some training is performed prior the actual trials, where the height of the P300 response is estimated for each subject. This article is relevant for this thesis since it proves that it is possible to detect the P300 just by peak detection.

Lu et *al.* [23] research on the impact ISI, target duration and ISI to target duration ratio has on the performance of the system. According to their study, higher ISI obtained the most accurate results. Besides, longer target duration also improved the accuracy. As for the ratio, experiments with high ratio also proved to be more accurate. As for the P300 waveform, greater ISI resulted in higher peak amplitude.

Finally and regarding possible features for P300 bases BCI systems, there are little but consistent studies on the use of a combination of wavelet decomposition and entropy [24][25] Perseh and Sharafat [25] decompose the EEG signal via Multi Resoution Analysis to retrieve the freqeuncy bands where the P300 response can be found.

Quiroga et *al.* [24] performed a study on the relationship between wavelet decomposition of EEG signals and its entropy. The results did show correlation between event-related oscillations and entropy variations.

4.3 Conclusions

The literature review showed that the two main paths of investigation in P300 based BCI systems are: adaptability of the system and interface design parameters impact on its performance.

The following chapter will go through the techniques that were developed for this thesis. Some of them are based on the state of the art here discussed.

Chapter 5

EEG Processing

As it was already described in the previous chapter, EEG processing is the central part of every BCI system.

In this chapter, the preprocessing techniques used in this project will be presented. The feature extraction methods researched and implemented for the project will also be described. The classification scheme followed will conclude the chapter.

It must be noticed that all of the algorithms hereby presented have been designed and implemented for a P300 based BCI system. Consequently, all of the following sections are based on this approach.

5.1 Preprocessing

5.1.1 Frequency Filtering

As seen in the first chapter, EEG activity takes place in a specific range of frequencies. In the same, the noise introduced by, for example power line interference, has also its specific frequency band of appearance. This situation can be taken as an advantage to improve the SNR of the EEG signal. Regarding ocular movement, its frequency range varies from 0.1 to 10 Hz.Consequently, a frequency filtering will not get rid of this artifact and other solutions must be sought. On the contrary, ECG has a wider frequency range (0.01 - 300 Hz). Therefore, a band pass filter will be effective in this case. This project is focused on the P300 wave, which is known to be more prominent in the lower frequency bands[24][18]. The selected band pass filter will be then a fourth-order Butterworth filter. Cutoff frequencies are set to 1 and 20 Hz. A Butterworth filter has flat frequency response within the band pass, which represents a highly desirable feature so as not to distort the EEG signal within the band of interest.

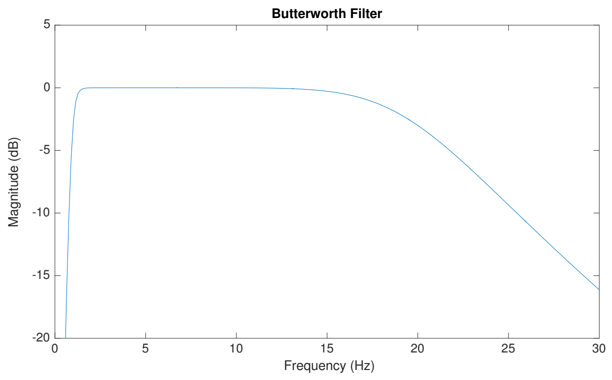


Figure 5.1: Magnitude Response of the Butterworth Bandpass Filter

As for the power line interference, it can be seen at 50 or 60 Hz, depending on the specific power installation of the environment the measurements are being taken at. In the concrete sceneario of this project, the power line interference acts at 50 Hz. To overcome this problem, a Notch filter with such cutoff frequency is used to filter the EEG signal. A Notch filter is a band stop filter with a highly narrow band stop frequency band. Therefore, it is highly effective when removing a specific frequency from a signal.

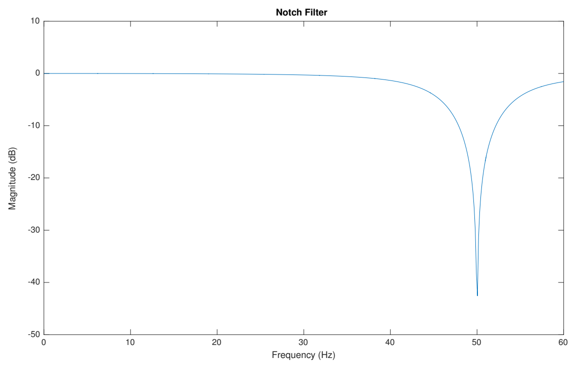


Figure 5.2: Magnitude Response of the 50 Hz Notch Filter

After the frequency filtering, the signals were standardized, i.e. substracted its mean and divided by their standard deviation.

5.1.2 Component Based Methods

These methods attempt to separate the artifacts from the EEG signal by identifying certain patterns in the data. In general, all these techniques project the original signal into a subspace that is capable of retaining more effectively the information of interest to the system. Consequently, Component Based Methods help reduce the computational requirements of the system, since less input data will be able to produce better results.

Principal Component Analysis

Principal Component Analysis (PCA) aims to reduce the dimension of data without much loss of information. The recorded signals are correlated with each other and, therefore, information coming from all channels can be redundant and decrease the classification performance[26].

PCA goal is to uncorrelate this set of vectors, maximizing the variance of the data. Consequently, the result will consist of a set of vectors that retain the most important information for the BCI system[27].

The PCA algorithm is now presented for an input signal X[28]:

1. Standardize the data
2. Perform Singular Value Decomposition (SVD)
3. Sort eigenvalues in descending order and choose the k eigenvectors corresponding to the k largest eigenvalues, being k the new dimension of the data subspace
4. Construct the projection matrix $\mathbf{W}$, which is just a concatenation of those largest eigenvectors.
5. Transform the original signal $\mathbf{X}$ via $\mathbf{W}$

This procedure is also known as prewhitening of the input data and can be regarded as a first step to a complete separation of vectors [29]

Independent Component Analysis

Independent Component Analysis (ICA), as PCA, is a technique that seeks the maximum independence of the input components. While PCA relies on second order statistics, i.e. covariance, ICA can be seen as a generalization of PCA which does not require orthogonality. To this purpose, ICA will find a transformation matrix that minimizes the mutual information between the different sources [27]. There are many interpretations of independence and, therefore, several ways of computing ICA can be found in the literature: Maximum Likelihood Estimation (ML), Entropy Maximization and Maximization of non-Gaussianity [29]. The last mentioned method is the most commonly seen in BCI systems.

The central limit theorem states that the summation of two or more statistically independent variables tend to have a Gaussian distribution but this summation becomes less Gaussian when it resembles one of those independent components. Consequently, maximizing the nonGaussianity of a summation of sources (each of the EEG channels in this project), we will be able to estimate one of them [29].

There are many ways for measuring nonGaussianity. The one used in this project is called Negentropy. Negentropy is a measure of the distance of a random variable from the Gaussian distribution [29]. It is the evident that maximizing nonGaussianity is equivalent to maximizing negentropy. Negentropy is based on the information measurement called entropy, which will be later explained in the "Feature Extraction" section.

Prior the ICA method, two procedures are usually followed: centering and whitening. Centering simply means to subtract the signal its mean. This is not a necessary step for ICA but it does simplify the method. Whitening obtains a new vector whose components are uncorrelated and have unity variance. The simplest way to perform whitening is the eigenvalue decomposition (EVD) of the covariance matrix of the input vector:

$$\hat{X} = ED^{-\frac{1}{2}}E^{T}X \tag{5.1}$$

Where D is the diagonal matrix of eigenvalues, E is the orthogonal matrix of eigenvectors and X is the input signal.

The ICA method will be implemented in this project in a more efficient way, called FastICA.

FastICA

FastICA [30] is a fixed-point algorithm to maximize nonGaussianity. This method has considerable advantages in comparison to the technique it is based on, ICA. First of all, the convergence of the algorithm is really fast and consequently suitable for online purposes. The algorithm does not rely on step size parameters. Therefore, it is easy to use and not susceptible to wrong criterion when selecting those parameters. Also, there is no need to estimate the probability density function (PDF) of the input data, as it is required for Negentropy based ICA. This is specially important in the field of BCI since EEG signals are far from having a established PDF.

In this project, the implemented FastICA for several units was implemented due to the fact that the input EEG signal comes from different channels. The procedure to be followed is identical to the algorithm for one unit or dimension, with a remarkable exception. To prevent the transformation vector components to converge to the same value, they need to be decorrelated after every iteration of the algorithm.

To measure nonGaussianity, FastICA relies on a non-quadratic non-linearity function $f(u)$, its first derivative $g(u)$ and its second derivative $g'(u)$. The non-linearity chosen is:

$$f(u) = -e^{-\frac{u^2}{2}},\ g(u) = ue^{-\frac{u^2}{2}},\ \text{and}\ g'(u) = (1-u^2)e^{-\frac{u^2}{2}} \tag{5.2}$$

FastICA algorithm steps are now presented, where $W \in \mathbb{R}^{n\times c}$, n being the number of dimensions of X and c the number of dimensions of the new subspace, is the transformation matrix and W_i is a column vector of this matrix, , which projects the input signal X into its principal components. This is the algorithm to be followed for every dimension [29], where $X \in \mathbb{R}^{n\times m}$, n being the number of dimensions and m the number of samples.

Algorithm 1 FastICA algorithm

```
1: Begin with a random initial normalized $W_i$
2: Update $W_{i+1} \leftarrow E\{Xg(w_i^T X)\} - E\{g'(w_i^T X)\}X$
3: Correct $W_{i+1} \leftarrow W_{i+1} - \sum_{j=1}^{p} W_i^T W_j W_j$
4: Normalize $W_{i+1}$
5: Go to step 3 until $W_i$ has converged
```

Convergence here means that W_{i+1} and W_i point in the same direction, i.e. their dot product is almost equal to 1.

5.1.3 Spatial Filtering

These processing techniques aims to reduce the dimensionality of data while improving the SNR by linearly combining the original data channels, obtaining a set of different pseudo-channels.

A wide variety of Spatial Filters (SFs) have been described in the literature. Among them, the most successful ones are the Common Spatial Patterns (CSP) filters, πSF or the xDAWN filter.

Some BCI approaches were based on prior training of the subject to achieve a high system performance. On the contrary, SFs focus on calibrating the system itself so as to adapt to each user [31].

This calibration does have some drawbacks though. These techniques are of course subject-dependent and therefore they need to acquire a certain degree of user-independence to achieve a good performance.

Besides, since many classification algorithms are dependent on the data set, if new data is included in the training data set, the whole classification will have to be computed once again. These kind of algorithms are called batch algorithms because they have to store the whole data set before training the classifier.

The need for adaptive SFs is then evident. If adaptive, not only the performance will be seen improved but also the speed and computational requirements of the system will be superior. Specially in the field of online BCI systems, adaptive SFs are of prior importance.

In this section two algorithms will be described. First, the xDAWN algorithm will be explained. Then, its adaptive and improved version, adaptive xDAWN (aX-DAWN) will be presented.

xDAWN

The basic idea behind the xDAWN is to enhance the synchronous response to target stimuli. The process can be then divided into two steps: estimate the synchronous responses and design spatial filters that enhance such potentials. The recorded EEG signal is then defined as:

$$X = (DA)^T + N \tag{5.3}$$

Where $X \in \mathbb{R}^{n\times m}$, n being the number of dimensions and m the number of samples, is the recorded signal. $A \in \mathbb{R}^{e\times n}$, e being the number of samples of the ERP and n the number of dimensions, represents the ERP response itself. $N \in \mathbb{R}^{n\times m}$, n being the number of dimensions and m the number of samples, stands for the noise and $D \in \mathbb{R}^{m\times e}$, m being the number of samples and e the number of ERP samples, is a Toeplitz matrix whose first column elements are all null except for the time instants where there is a stimulus onset.

Minimizing the least square error between the recorded EEG signal, X, and the product of D a the ERP response, A, will yield an estimation of this last signal.

$$\hat{A} = \arg\min_{A} ||X - (DA)^T||_2^2 \tag{5.4}$$

However, the least squares estimation gives a very redundant solution and, therefore, another scheme must be followed. The technique chosen is precisely the idea of a set of spatial filters, enhancing the synchronous responses: $U \in \mathbb{R}^{n\times c}$, n being the number of original dimensions and c the number of dimension of the projection. **ui** then represents each spatial filter, i.e. each column of U.

$$(X)^T U = DAU + (N)^T U \tag{5.5}$$

The first approach is to perform a PCA analysis of the estimated A and then project the signal X on the largest singular values of the Nf filters.

Since the noise is not directly taken into account when estimating the spatial filters, this estimation has to be further improved to increase the SNR.

The final solution will go through the computation of the QR decomposition of both D and X and a singular value decomposition.

The QR decomposition decomposes a given signal into an orthogonal matrix, Q, and a triangular matrix, R. It is a commonly method used to solve least squares problems. Being $Q_D \in \mathbb{R}^{m \times e}$, $R_D \in \mathbb{R}^{e \times e}$, $Q_X \in \mathbb{R}^{m \times n}$ and $R_X \in \mathbb{R}^{n \times n}$.

The *Nf* largest eigenvalues of the matrix formed by the Q segments of both matrices will be later selected.

$$Q_D^T Q_X = \Phi \Lambda \Psi^T \tag{5.6}$$

Φ and Ψ represent the matrices of left and right singular values (ϕ_i and ψ_i), respectively, and Λ the matrix of singular values, λ_i.

The xDAWN algorithm will be now summarized in six steps to estimate the evoked subspace.

Algorithm 2 xDAWN algorithm to estimate evoked subspace

```
1: Compute QR factorisation of $X \Rightarrow X = Q_X R_X$
2: Compute QR factorisation of $D \Rightarrow D = Q_D R_D$
3: Compute SVD of $Q_D^T Q_X \Rightarrow Q_D^T Q_X = \Phi \Lambda \Psi^T$
4: Select the $I$ couples of singular vectors $(\Phi_i \Psi_i)$ associated with the $I$ largest
   singular values $\lambda_i$
5: Finally $\forall\ 1 \leq i \leq I$, $(u_i, a_i') = (R_X^- 1 \psi_i, R_D^- 1 \phi_i \lambda_i)$
6: Estimate sources $\forall\ 1 \leq i \leq I$, $\hat{x}_i(n) = \hat{\mathrm{u}}_i^T x(n)$
```

Where a_i' stands for each column of the ERP matrix, A and n is the time index.

aXDAWN

The xDAWN algorithm improves its predecessor, xDAWN, by including a recursive least squares (RLS) adaptation of the spatial filter coefficients. The RLS method not only allows to introduce new data into the training set, but also change the spatial filter coefficients upon changing conditions on the input EEG signal.

Being performed on each incoming window of data, the computational complexity and memory requirements are obviously reduced in comparison to the previously described method.

The aXDAWN introduces the temporal dimension, and iteratively obtains the spatial coefficients by maximizing the current state of the signal-to-signal-plus-noise-ratio using generalized eigenvalue decomposition (GED)

$$\hat{\mathbf{U}}(n) = \arg\max_{W(n)} \frac{U(n)^T R_1^1(n) U(n)}{U(n)^T R_2^1(n) U(n)} \tag{5.7}$$

Where the estimate uses both the current autocorrelation of the signal estimation $R_1^1(n) = \hat{A}(n)^T \hat{D}(n)^T \hat{D}(n) \hat{A}(n)$, $R_1^1 \in \mathbb{R}^{n \times n}$, and the autocorrelation of the noise-plus-signal estimation $R_2^1(n) = X(n)^T X(n)$, $R_2^1 \in \mathbb{R}^{n \times n}$. Each column of the estimated spatial filter, $\hat{u}_i$ will be then computed following the equation:

$$\hat{\mathbf{u}}_i(n) = \frac{u_i^T(n-1) R_1^i(n) u_i(n-1)}{u_i^T(n-1) R_2^i(n) u_i(n-1)} \cdot R_2^i(n)^{-1} R_1^i(n) u_i(n-1) \tag{5.8}$$

For numerical reasons, in each iteration these coefficients must be normalized. Moreover, to obtain similar values to those extracted from the regular xDAWN algorithm, the final estimation of each one of the spatial filters will be subtracted from the following scaling function:

$$\mathbf{S} = diag(\hat{U}^T R_2^1 \hat{U}) \tag{5.9}$$

$$u_i \leftarrow \sqrt{\frac{1}{s_{ii}}} \hat{u}_i \tag{5.10}$$

Where

$$s_{ii} = diag(\hat{u}_i^T R_2^i \hat{u}_i) \tag{5.11}$$

Since it is not feasible to invert the noise autocorrelation in every step, an alternative method following the Sherman-Morrison-Woodbury formula [19].

On its behalf, the lower order filters will be calculated using a deflation technique in order for them to be uncorrelated.

$$R_1^i(n) = \left[I - \frac{R_1^{i-1}(n) u_{i-1}(n) u_{i-1}^T(n)}{u_{i-1}^T(n) R_1^{i-1}(n) u_{i-1}(n)} \right] \cdot R_1^{i-1}(n) \tag{5.12}$$

$$R_2^i(n) = R_2^{i-1}(n) \tag{5.13}$$

Finally, A needs to be estimated. If the ERPs are not overlapped, the solution would be as simple as an averaging procedure. However, this condition is not always met. Again, the Sherman-Morrison-Woodbury [19] formula will be used, combined with the mentioned averaging procedure. Due to the adaptive nature of the filter being described, this last procedure will be performed iteratively as the other steps previously presented.

In order to adapt the filters, there are two different possibilities of action: tracking the ERP response or the noise autocorrelation. In this project, the first option was chosen and, thereby, the presented formulae to be followed will only be the ones corresponding to such method.

$$\hat{A}(n) = \lambda_A \hat{A}(n-1) + \frac{\hat{X}(n) - \hat{A}(n)}{k} \tag{5.14}$$

$$R_1^1(n) = \hat{A}(n)\hat{A}^T(n) \tag{5.15}$$

Where k is the k the ERP and $\lambda_A \in [0,1]$ is the update coefficient and I is the identity.

The raXDAWN [21] variation introduced in the literature review was implemented in the system. The only difference between both method is the initialization:

- $R_1^1(0) = 2^{-15} I$
- $R_2^1(0)^{-1} = 2^{15} I$

The initialization parameter was chosen after several experimental trials.

The complete algorithm is now stated.

Algorithm 3 The aXDAWN spatial filter

```
Initialization: $\hat{U}(0) \leftarrow$ small random numbers
$\hat{A}(0), R_1^1(0), R_2^1(0) \leftarrow 0$ $R_2^1(0)^{-1} \leftarrow I$
1: begin
2: while more data available do
3:   Receive data window $X(n)$
4:   if $X(n)$"$target$" then
5:     Update $\hat{A}(n)$
6:     Update $R_1^1(n)$
7:   end if
8:   Update $R_2^1(n)$
9:   Update $R_2^1(n)^{-1}$
10:  for $i \in [1...n_f]$ do
11:    if $i > 1$ then
12:      Apply deflation technique to get $R_1^i(n)$
13:    end if
14:    $u_i(n-1) \leftarrow$ column of $\hat{U}(n-1)$
15:    Calculate $u_i(n)$ using the RLS update
16:    Normalize $u_i(n)$
17:    Save $u_i(n)$ in $\hat{U}(n)$
18:  end for
19:  Get $U(n)$ by de-normalizing $\hat{U}(n)$
20: end while
21: end
```

5.2 Feature Extraction

Once the EEG signal has been appropriately preprocessed, the next step to follow is to perform an adequate feature extraction that reflects the information sought in the EEG. This section will cover the methods used in this projects.

5.2.1 Time - Frequency

Spectral methods, such as Fourier Transform (FT) or Short Time Fourier Transform (STFT), have been widely used in signal processing due to its simplicity and computational speed. These techniques, however, imply important disadvantages in the analysis of ERP signals. FT yields a representation of the signal with perfect spectral resolution but retains no temporal information.

STFT, a Fourier-related transform commonly used in the analysis of non-stationary signals, demands a constant time windowing of the signal, which eventually affects frequency resolution [27].

Wavelet transform (WT), on the contrary, is local in both time and frequency and therefore it allows a thorough analysis of the time-varying spectral content of the signals. WT distinguishes between continuous (CWT) and discrete (DWT) mode. CWT can be a highly redundant representation of the analyzed signal. Besides, it is computationally impossible to use the whole infinite set of wavelet coefficients. Consequently, a discrete set of parameters can be selected and perform a DWT instead of a CWT. The result is a simpler and faster operation, which can also be implemented via digital filtering techniques.

The basic idea behind DWT is a projection of the input signal into different basis functions built from translations and scaling of a set of functions called mother wavelets (MVs). By definition, mother wavelets are finite energy, fast decay signals whose time-width adapts to their frequency: high frequency wavelets correspond to narrow functions while low frequency are much broader. Different resolutions are achieved using both scaling and mother wavelet functions. The scaling and mother wavelet functions are defined, respectively, in [32] as :

$$\phi(X) = \sum_{k=-N}^{N} a_k \phi(2X - k) \tag{5.16}$$

$$\psi(X) := \sum_{k=-N}^{N} (-1)^k a_{1-k} \phi(2X - k) \tag{5.17}$$

Where N is the number of samples.

Multi Resolution Analysis

Multi Resolution Analysis (MRA) is an efficient tool to analyze the time-frequency content of a given signal in different resolutions and scales. Multiresolution consists of a series of nested approximation spaces, V.

All of these spaces are scaled versions of one space, which reflects precisely the multiresolution aspect of the wavelet transform [32].

MRA can be easily implemented via dyadic filter bank. Two filter and two down samplers compose each decomposition stage. A mother wavelet and a scaling function represent the two filters: high-pass and low-pass, respectively. The process is quite simple; the input signal is low-pass filtered to obtain the approximation coefficients and high-pass filtered to obtain the detail coefficients. Then both output signals, detail and approximation, are down sampled. The decomposition further continues by taking the approximation coefficients as the input signal for the next stage.

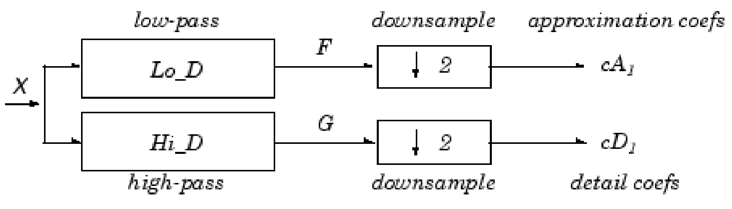


Figure 5.3: MRA Block Diagram (Matlab)

The most important steps in a MRA process are the proper selection of the mother wavelet and decomposition level. The choice of the mother wavelet will be highly dependent on the sought feature. Daubechies MVs are characterized by their extreme phase and highest number of vanishing moments. Consequently, they are a great choice when smoothness is required. On their behalf, Shannon MVs are analytically defined, infinitely differentiable and sharply bounded in the frequency domain. These properties provide pretty good location of energy in the spectral domain. It has been widely reported in the literature that Daubechies 4 is the best choice to detect changes in EEG signals. Moreover, this function resembles the P300 component in ERPs so they make the perfect and adequate option for this project [25] .

The number of decomposition levels will determine which frequency sub-bands are retained in the wavelet coefficients. So as to retrieve all the effective frequency components in the ERPs, a sufficient number of levels must be computed. The frequency scale of the k detail decomposition levels follows the pattern:

- $f_k = [0, \frac{f_s}{2^k}]$

- $f_{k-n} = [\frac{f_s}{2^{(k-n)}}, \frac{f_s}{2^{(k-(n+1))}}]$

Where f_s is the sampling frequency.

The most prominent component of the P300 response has been found in the delta frequency range and, consequently, the highest decomposition level must contain frequencies within 0 and 4 Hz. Given the sampling frequency of 256 Hz, both of the amplifier used in this project experiment and in the offline data tested, the appropriate number of decomposition stages is six. The resulting sub-bands correspond then to the following frequencies:

Frequency Range	Decomposition Level
0 - 4 Hz	6
4 - 8 Hz	6 - Approximation
8 - 16 Hz	5
16 - 32 Hz	4
32 - 64 Hz	3
64 - 128 Hz	2
128 - 256 Hz	2

Table 5.1: MRA Decomposition Levels

All frequencies, except explicitly stated, correspond to the detail level.

Each decomposition level represents the same temporal interval but, due to the downsampling performed in the decomposition, the higher the level, the less coefficients a given level has. As a consequence, the length of the input signal to be decomposed has to be wisely chosen. The reason is that further operations on the wavelet decomposition will have, at most, a time resolution equal to that of each coefficient of the highest decomposition level. Being the timing extremely important in a P300 BCI system, the MRA was computed each 1.5 seconds so as to obtain a time resolution of 125 ms.

5.2.2 Entropy

In the field on Information Theory, entropy was first introduced by Shannon [33]. Entropy is a measurement of the amount of information retained in any time series. Regarding information theory, entropy is based on the PDF of such series [34]. With this measurement, Shannon introduced a way of measuring the uncertainty. Consequently, it can also be regarded as a measure of the spread of the data. Flat PDFs will have high entropy values will narrow PDFs will have low entropy values [35]. The reason is obvious: a narrow PDF indicates that all of the possible values a time series can present are focused on a small range of a values. Therefore, there is not much uncertainty and the entropy will be low.

Entropy can also be regarded as a measure of disorder and chaos. The more chaotic a distribution is, the higher its entropy. Applied to the field of BCI systems, it should be expected that EEG signals have high entropy values. Nevertheless, the P300 response can be regarded as a transition from a disordered state in the brain signal to an ordered state. In this case, the entropy will be subject to decrease when this response is present.

Once discussed the potential importance of entropy in BCI systems, this measurement will be now explained.

First of all, entropy, from now on being referred to as H was described as a measurement with the following main properties [34]:

1. H should be a continuous function
2. If all n observations in a time series have equal probabilities, then H should be a monotonic increasing function of the number of n
3. The entropy of a summation of two independent probability distributions should be the sum of the individual entropy of each distribution

The function that satisfies such conditions is [34]:

$$H(X) = -\sum_k p(X_k)\log_2(p(X_k)) \tag{5.18}$$

Where $p(X_k)$ is the probability of each observation in a time series X_k.

Renyi proposed another family of entropy measurements based on Shannon's word [34]. Shannon's is the simplest of all entropy measures and it is considered to be the averaged information in the usual sense[34]. Renyi's entropy is defined as:

$$H_\alpha(X) = \frac{1}{1-\alpha}\log_2(\sum_k p(X_k^\alpha)) \tag{5.19}$$

The placement of the α makes Renyi's entropy more flexible than Shannon's since it enables several measurements of uncertainty for a given PDF. In the case where α equals one, Renyi's entropy is equivalent to Shannon's.

The α power allows to control the contribution of the tails of the PDF to the entropy. This is important, for example, in the case that weak signal components overlapped with stronger ones must be distinguished from the latter.

A large positive value of α will provide a measure much more sensitive to events that occur often. On the contrary, large negative values will give measurements more sensitive to events which happen seldom.

In this project, the entropy measurement chosen is Quadratic Renyi's Entropy, $\alpha = 2$

Once the concept of entropy is understood, the next section will describe how both the ideas of wavelet decomposition and entropy are combined in this project as a feature.

5.2.3 Wavelet and Entropy

As discussed in the previous section, entropy can reveal ordered states in EEG signals. Also, MRA provides a good time-frequency analysis of the input signal. Consequently, a combination the concepts of MRA and entropy can be used as a feature to characterize the state of EEG signal and determine if the P300 response is present or not.

Entropy is fairly easy to compute, provided the probability density function of the signal under consideration is known. Of course, EEG signals are non-stationary and no PDF fits them appropriately. Therefore, it is not appropriate to estimate entropy with a parametric PDF model and it is required to choose a non-parametric estimation.

The first method attempted was entropy estimation via kernel function. The kernel (Parzen) estimate of the PDF using an arbitrary function $\kappa_\sigma(.)$ is given by [34]:

$$\hat{p}_X(X) = \frac{1}{N_\sigma}\sum_{i=1}^{N}\kappa\Big(\frac{X-X_i}{\sigma}\Big) \tag{5.20}$$

where N is the number of independent variables and σ is the size of the kernel size or bandwidth parameter. The kernel size affects the bias and the variance of the estimation in opposite ways. Therefore, it should be wisely selected.

A Gaussian function was chosen as kernel function due to its ease of implementation. Using eq. (5.19), the estimation of the quadratic Renyi entropy will be followed as[34]:

$$\hat{H}_2(X) = -\log\int_{-\infty}^{\infty}\Big(\frac{1}{N}\sum_{i=1}^{N}G_\sigma(X-X_i)\Big)^2 dx \tag{5.21}$$

Where G_σ is the Gaussian kernel function and $X(n)$ is the input signal data.

The result of this operation is easily computed since the product of two Gaussians can be exactly evaluated as a Gaussian computed at the difference of the arguments and whose variance is the sum of the variances of the two original Gaussian functions [34]:

$$\hat{H}_2(X) = -\log\sum_{i=1}^{N}\sum_{j=1}^{N}\Big(\frac{1}{N^2}G_{\sigma\sqrt{2}}(X_j-X_i)\Big)^2 dx \tag{5.22}$$

With this method of entropy estimation, the step of estimating the PDF of the input data has been avoided. It is widely considered as a good decision to choose this parameter accordingly to the dimension of the data, following Silverman's rule [34]:

$$\sigma_{opt} = \sigma_X (4N^{-1}(2d+1)^{-1})^{\frac{1}{(d+4)}} \tag{5.23}$$

Where σ_X is the standard deviation of the input data, N is the number of samples and d is the dimensionality of the data.

This process, although accurate, was time consuming and presented too high computational requirements. Consequently, this approach needed to be discarded since BCI systems require accuracy but also speed.

Accordingly, wavelet spectral entropy was the selected feature to be used. Spectral entropy quantifies the spectral complexity of an input signal[35], EEG in this project. The basic idea behind spectral entropy is to compute this value from the PDF of each wavelet decomposition level. This PDF is defined as the ratio of each level's energy to the total mean energy.

On its behalf, the energy for each level was obtained by squaring its wavelet coefficients. Some relevant facts must be taken into consideration before proceeding to calculate this energy.

As discussed, each resolution level has a different number of coefficients and, as a consequences, what must be computed is the mean energy in the decomposition level rather that its total value. The mean energy will be computed by averaging the total energy over a time window long enough to include at least one coefficient in every level [24]:

$$E_k = \frac{1}{N} \sum_i C_{i,k}^2 \tag{5.24}$$

Where E_k is the energy of each decomposition level in each window, $C_{i,k}$ the wavelet coefficients and N is the number of wavelet coefficients in that specific window.

The total energy for each time window will be then:

$$E_{tot} = \sum_k E_k \tag{5.25}$$

The PDF of each level will simply be computed as the ratio of each decomposition level energy to the total energy:

$$p_k = \frac{E_k}{E_{tot}} \tag{5.26}$$

Finally, Renyi's entropy can now be calculated with eq.(5.19)

Of course, this method is less accurate than the previous one. Nevertheless, the computational requirements are considerably decreased, achieving a faster and enough accurate method for this project purposes.

5.3 Classification

Once the feature extraction has been completed, this information must be processed by a classifier. The aim of the classifier is to find, as accurately as possible, the P300 response elicited by the low-probability target event.

5.3.1 Peak Detection

Peak Detection is the first classification algorithm attempted in this thesis. Both adaptive and non-adaptive schemes have been developed. The peak detection for both of them are similar. This subsection will first explain the features used in both techniques, and lastly the differences in the implementation will be presented.

Apart from the computed wavelet entropy which can reveal ordered states in EEG signals, i.e. P300 responses, this ERP is shown as an increase in the delta band power. It is then expected to have significant peaks in that frequency subband.

Consequently, the selected features for this project will be both the wavelet entropy and the delta energy.

The wavelet entropy is expected to rapidly decrease on the appearance of a stimulus. Although the entropy will be constantly varying due to the non-stationary nature of the EEG signal, this variation will be approximately constant. Therefore, until the presence of an ERP, the variance should be kept more or less stable. However, during such stimulus episodes, the entropy will find its minimum value and, consequently, the entropy variance will be maximum [36]. The first feature will be then the variance of the wavelet entropy, compute over consecutive time windows (these windows being the minimum time resolution level of the MRA)

The other feature will not consist only on the delta energy itself. The relation between the delta increase and the wavelet decrease must be taken into account since an increase in the energy of this frequency band can be caused not only because of the presence of an external target stimulus. To compute this feature, first, the entropy will be substracted its maximum value. Next, its absolute value will be taken so as to convert the previous local minimum to local maxima. Finally, the energy will be scaled by this value. In this way, the energy increments will now coincide with the entropy decrements.

Due to the nature of the computed features, the classifying decision will be solely based on peak detection. Each of the obtained features will undergo a peak analysis, where the threshold will be adaptive for each of them. Consequently, this threshold will be adaptive for each trial and each user. The reasons for the threshold to be adaptive are quite obvious. Although the features are all normalized, their shape are completely dissimilar and some of them need more strict values as a threshold. That is the case of the entropy, for example. Regarding the adaptivity between trials and users, the stimulus sequence changes in every trial so it cannot be expected that the brain signals will behave similarly, less between different subjects.

Based on experimental procedure, the threshold selected for the entropy variance equals its mean. As for the energy feature Also, consecutive peaks cannot be closer than the ISI.

Knowing the interval in which the P300 response can appear, 200-400 ms, every row/column that has appeared from 125 to 375 ms prior the peak will be considered as target stimulus.

The classification procedure for the aXDAWN filter is slightly different. Since in each iteration the filter needs to know if the input signal corresponds to an ERP or not, the classification must be computed for each input segment of the filter. First, the considered threshold is zero since in such small time windows there is not enough data to select an appropriate threshold.

Also, instead of considering every peak higher than the threshold to correspond to an ERP response, the classification algorithm for aXDAWN only considers as target those peaks found both in the entropy and energy feature. This ensures the condition that a decrease in the entropy must be followed by an increase in the energy. The position of this peak will be saved and, in order to minimize errors, the same procedure followed for the non-adaptive methods is followed: rows and columns appearing from 125 to 375 ms before the ERP are considered target.

Due to the interval appearance of the P300 wave, it is quite easy to choose a non-target row/column as stimulus. Although, the stimulus sequence is randomized, this variation led to many misclassifications and, as a consequence, the classification method was changed.

5.3.2 Support Vector Machine

This classifier uses a discriminant hyperplane which maximizes the margins between different types of data. Non-linear decision boundaries can be created with only a low increase of complexity by mapping the data to another space of much higher dimensionality. The pros of these classifiers are their good generalization properties and insensitivity to overtraining. However, they suffer from low speed of execution[5][37].

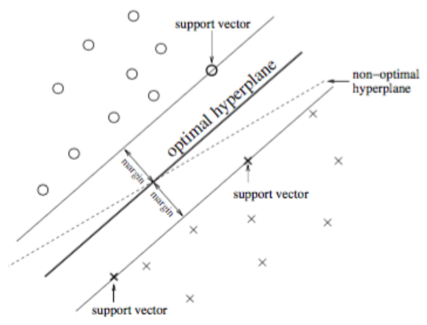


Figure 5.4: SVM finds the optimal hyperplane for generalization [5]

Given the results obtained from the previous classifier, which will be later presented in the following chapter, it was decided to change the classification scheme and implement a more complex algorithm.

Although the combination of wavelet and entropy is able to highlight the presence of the P300 response, there is a huge problem with its delay of appearance. Since the ISI is similar to the stimulus duration, consecutive rows and columns are too close to each other. Consequently, a slight variation in the time interval where the P300 response is being searched can highly affect the performance of the system. If the P300 wave had a constant delay after the presence of the stimulus, the peak detection classifier would work almost perfectly. However, this is not the case and other classifier had to be implemented.

The previously described properties of the SVM classifier made it the selected next classifier. The features used for this classifier are the same two ones previously described, plus one other feature that was found to show the presence of the P300 response.

This other feauture is simply the inverse of the wavelet entropy. Huge decreases in this time series will be seen as huge peaks in the inverse signal.

The training set of the classifier consists then of the features:

1. Entropy variance over consecutive samples
2. Entropy scaled by delta energy
3. Inverse of the entropy

After a set of trials, the selected features for the SVM are here presented:

1. **Kernel Function**: Gaussian
2. **Box Constraint** : parameter controlling the penalty imposed on margin-violation observations - the higher, the longer training time - 5
3. **Kernel Scale** : 20

In the context of the P300 Speller, the classifier was trained in every case for 10% of the duration of a single letter trial.

5.3.3 Voting Process

The final step of the classification consists of a voting process. The results of every trial will be analyzed and the most frequent one will be selected as the target stimulus.

Each row and column of the matrix has been labeled from 1 to 12. Indices from 1 to 6 represent columns and indices from 7 to 12 rows. As it has been previously explained, the output of the classifier is a series composed by those indices that have been considered to be target stimulus. Being the test a sucesion of trials, by the end of the test the classifier will have obtained multiple outputs. Some of them will be correct but some of them will not.

The mentioned voting process has to be performed in two steps. This is necessary because both the correct row and column index are needed so as to extract the character the subject had in mind. Therefore, the classifier cannot give two row indices or two column indices as an output.

Following this reasoning, the voting will be first done for the rows and then for the columns, or viceversa.

The voting process is quite simple. The classifier will have a count on how many times a row or a column has been selected as a target. Once all the trial have undergone the analysis, the row or column with higher score will be chosen as the target.

Finally, the target letter will be the intersection of the guessed row and column.

5.4 Conclusion

This chapter presented the processing techniques used in this thesis for each of the BCI system stages. Next chapter will present the experimental procedure that was designed to test these algorithms.

Chapter 6

Experimental Setup

In order to assess the validity of the algorithms presented in this thesis, they ought to be tested. For this purpose, an interface was developed and an experiment was conducted on several subjects. This chapter will describe the interface, its developing process and the experimental procedure that was followed.

6.1 P300 Speller

The paradigm chosen to test the presented algorithms is the P300 Speller. The speller is a simple system based on visual stimuli that allows a subject to spell a letter, consequently being able to form words and communicate with the external word simply by thought. Therefore, the speller is a potential communication tool, specially useful for people suffering from disabilities such as ALS.

The interface consists of a 6x6 matrix, containing both letters and numbers. While functioning, each row and column is randomly intensified, leaving a small time between those occurrences. The time interval between two consecutive row or column intensifications is called Inter Stimulus Interval (ISI) and it can have a high impact in the system's performance [9]. Within such random sequence of highlights, the appearance of the letter the subject is thinking about will represent a low-probability event. This situation will consequently trigger a P300 response, which will be recognized by the system. Then, the intersection of the ERP wave elicited by a row and the one stimulated by a column will reveal which letter the user was focusing on[37].

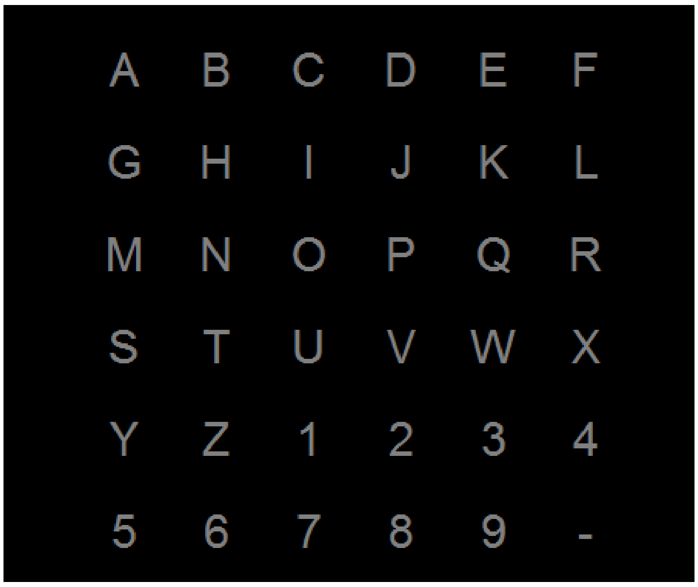


Figure 6.1: P300 Speller

6.2 Design of the Interface

To be able to present a visual stimulus, an appropriate toolbox had to be chosen. Being the paradigm based on visual stimulus, it was crucial to achieve synchronization between the images presented on the screen and its refresh rate. The toolbox selected for this purpose was the Cogent Graphics toolbox for MATLAB. This toolbox allows the programmer to draw different images and display them when desired on the screen. The basic command that makes this images, called ‘sprites’ by the toolbox, is ‘cgflip’ and it waits until displaying the picture fits the monitor refresh rate. Four different screens were needed for this project. All of them contained the mentioned matrix but each of them had a slightly different purpose. In order to synchronize the recording with the beginning of the interface, the first screen presented showed the matrix along with the text message “Press space bar to start”. The interface will then enter in a loop until that key is pressed. Then, the second sprite, consisting solely on the matrix, appears and the random order of the row and columns will be extracted via the coded ‘speller’ function. After some seconds, the sprite changes again. The remaining two sprites exhibit the matrix and the word to be spelled, followed by the specific letter the subject is supposed to be focusing on. One of these sprites will be static and change only to display the following letter and/or word. The other one will also do so, but in addition it will be the one in charge of changing the row and column colours that represent the above mentioned intensifications. The timing of the interface is controlled by the built-in MATLAB functions ‘tic’ and ‘toc’, which provided the most accurate timing results. Once the trial is over, the interface will return as an output the random sequence the matrix has followed, which is critical for the system, both to classify the P300 responses and to check the algorithms' results.

Being the matrix fixed, the main changes to be introduced in the interface are the following:

- Intensification time of each row or column
- ISI
- Number of words
- Letters in each word
- Time interval between letters
- Time interval between words

The selection of these parameters will be later explained in the 'Experiment' section.

6.3 Recording

The EEG recordings were performed with the g.tec amplifier. The specific hardware tool used was the amplifier g.USBamp, along with the included MATLAB API that allows the recording from MATLAB.

Figure 6.2: g.USBamp (g.tec Technologies)

Besides, the g.GAMMAsys provided the 16 electrodes used for the measurements and the g.GAMMAbox was the power supply and driver that connected such electrodes to the amplifier.

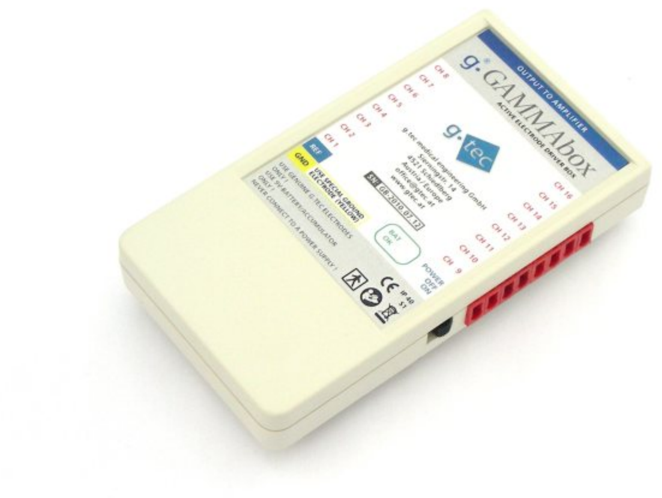


Figure 6.3: g.GAMMAsys (g.tec Technologies)

The electrodes placement followed the 10-20 system and they were chosen as follows: Fc3, P0z, Fz, Fc2, Fc4, P3, P08, Cz, C2, C4, Oz, P07, Cpz, Cp2, P4, Pz. The ground was placed on the forehead and the reference on the right ear lobe.

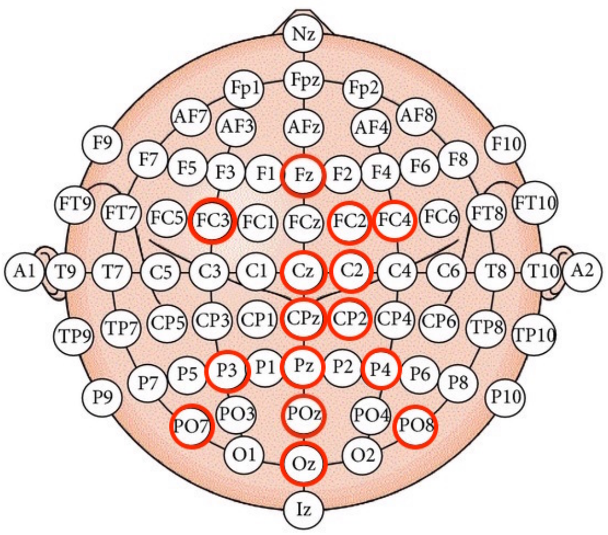


Figure 6.4: Electrodes placement for the experiment

To perform the measurements, a MATLAB script was coded. This script first initializes the amplifier, selecting 16 channels and a sampling frequency of 256 Hz. Synchronization between the interface and the recording script is crucial. Therefore, a User Datagram Protocol (UDP) connection is established between the recording and the interface scripts and both scripts are running in different MATLAB sessions. Just before initializing the interface itself, each interface scripts sends the duration of the trial to the recording script. Then, the recording will wait until the space bar key is pressed. The seconds between this occurrence and the appearance of the first word, concretely five seconds, will be discarded and the real EEG recording will start and be stored after this time interval. This time period is necessary mainly for two reasons. First of them is to assure a correct synchronization between recording and interface. The second one is to allow the EEG recording to stabilize, since the first recorded samples can be quite inaccurate due to the setup time.

6.4 Experiment

The experiment was carried out with 10 healthy voluntary subjects aged from 20 to 26 (5 females and 5 males) without any neurological disorder known to affect the EEG recording. One subject was left-handed and six out of the ten subjects described to have any of the following optical defects: myopia, astigmatism. The trials were conducted in an isolated and barely illuminated room so as to assure the quality of the recordings. The conducted experiment consisted of three different trials. In every one of them, each subject was facing the speller matrix, sitting comfortably approximately one meter apart from the screen.

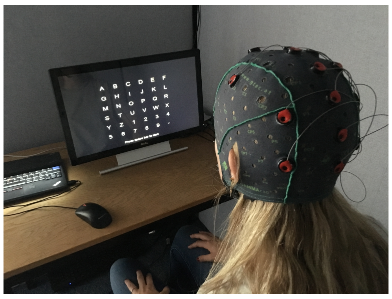


Figure 6.5: Subject using the speller

The speller presented predefined five-letter words in every one of the trials. For each letter, each row and column was randomly intensified 10 times for 125 ms, being then each element in the matrix highlighted 20 times. Every element in the matrix should be highlighted a sufficient amount of times so as to assure the system's performance, but that number should also be kept below a reasonable value. The reason for that last assumption is that the P300 wave is based on low-probability events and the sight of the desired letter for an excessive number of times could make the brain used to it, consequently not triggering the P300 response anymore. The chosen number of letters for each word was selected as five so as to obtain enough data to test the algorithms. As for the ISI, two different values were tested throughout the experiment. The first two trials kept the same ISI value: 250 ms.

Consequently, the interval between row or column intensifications was 125 ms. The third and last trial however had an ISI of 375 ms, leaving 250 ms between row or column highlights. The first impact that this change could have in the system could be an improvement in the classification. The P300 response has no fixed delay from the stimulus, ranging typically from 200 to 400 ms. Given the classification algorithm used in the system, a higher interval between row or column intensifications could lead to less misclassifications. On the other hand, increasing the ISI also increments the trial duration, which can have a negative effect on the subject.

The first trial, presenting only two words ("juice" and "clock"), was regarded as a test experiment to make the subject used to the system. In this way, the following and longer trials could be conducted with the certainty that the user really knew how to use the speller, enhancing then the quality of the recordings. The second trial included one more word ("phone", "eight" and "shoes"), increasing then the amount of data available to test the algorithms. The third trial ("mouse" and "beach"), being increased the duration of each word due to the ISI increment, presented only two words to the subject.

As a general feedback from the subjects, the least comfortable trial was the second one, most likely due to its higher duration. As for the most convenient, opinions were divided. Some of them preferred the first one, probably because it was the shortest trial, and others liked better the last one. The reason to this last choice was that the higher ISI made the trial easier on their eyes.

Finally, the time interval between consecutive letters was three seconds and the resting time between words was five seconds. It was considered that three seconds was enough for the subject to focus his or her attention on a different letter. Nevertheless, a new word could require longer adaptation time and so the time interval between words was chosen to be five seconds.

6.5 Conclusion

This chapter exposed the design for the P300 based BCI system that was implemented in this thesis. The following chapter will present the results obtained from this setup and other offline data.

Chapter 7

Results

This section will present the results obtained from this thesis. The algorithms were first tested with existing data sets from BNCI Horizon 2020. Once the experiment was conducted, the techniques developed were also tested in the data recorder specifically for the project.

7.1 Wavelet and Entropy

As it was discussed in the previous chapter, the delta energy and the wavelet entropy interact with each other at the presence of the P300 wave.

First, the contribution of each individual decomposition level was examined.

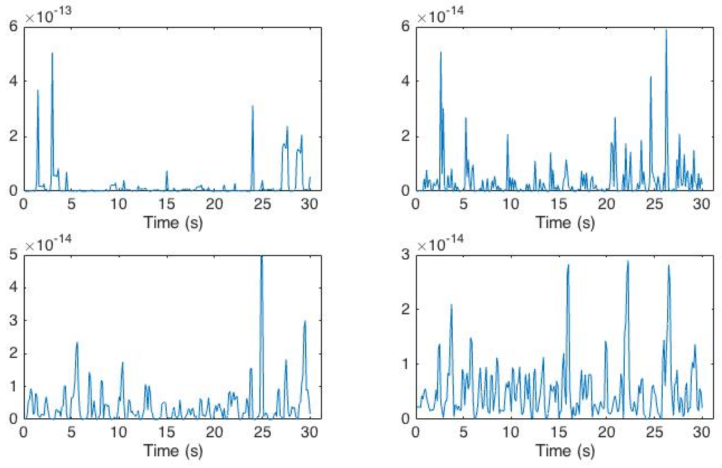


Figure 7.1: Energy within each decomposition level: (a) Delta Band - Detail; (b) Theta Band; (c) Alpha Band; (d) Beta band

The above figure presents only the lowest 4 decomposition level, ranging from 0 to 32 Hz, since EEG activity, specially during a cognitive task eliciting a P300 wave, takes place in that spectrum range.

The theoretical assumption about the predominance of the delta band during an external stimulus is then proved to be correct: the order of the delta energy is one order superior to the other levels.

This contribution can also be seen in the PDF of the wavelet. Since the delta level retains most of the energy, being computed the PDF as the ratio of each level power to the total power, it is expected that this level also has higher PDF.

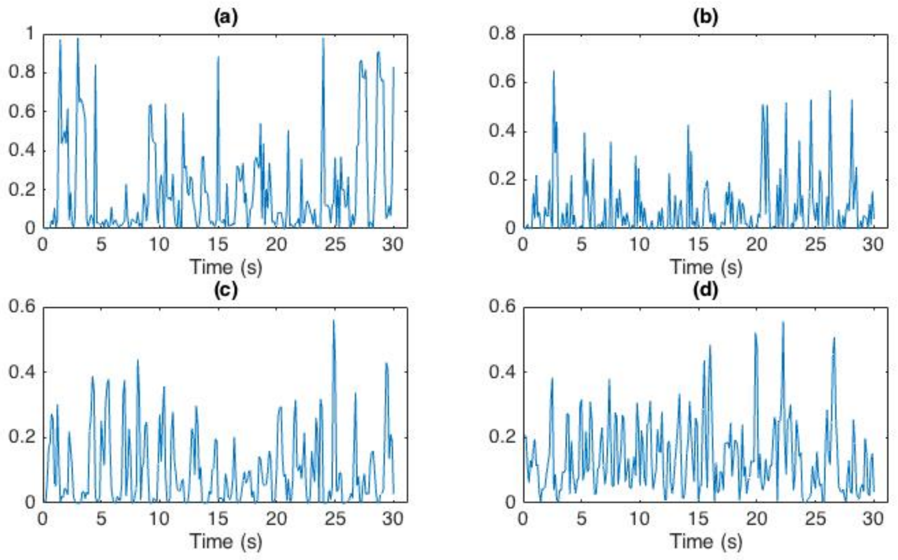


Figure 7.2: PDF of each decomposition level: (a) Delta Band - Detail; (b) Theta Band; (c) Alpha Band; (d) Beta band

Next, Renyi's entropy was computed. The figure below depicts the entropy value over a time window of 30 seconds, were several target stimuli took place.

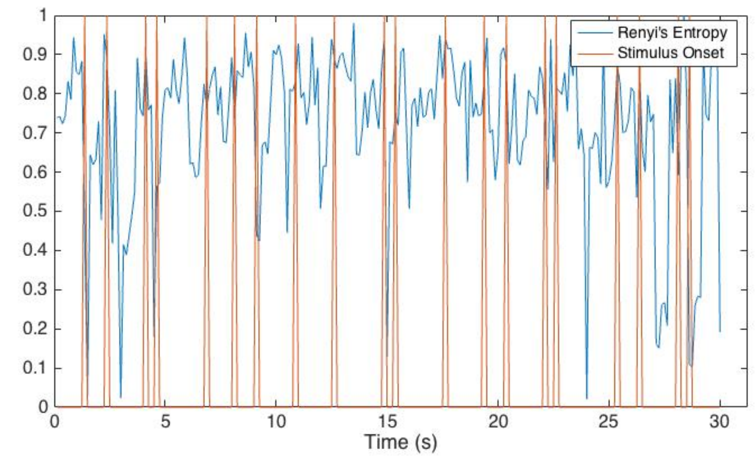


Figure 7.3: Renyi's Entropy and Stimulus Onset

Focusing on the first 3.5 seconds of the previous figure, the entropy decrease due to the stimulus is evident.

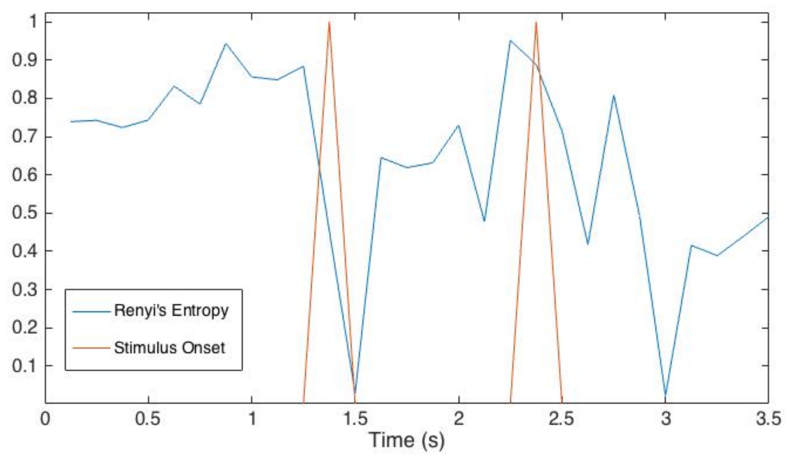


Figure 7.4: Zoomed view of Renyi's Entropy and Stimulus Onset

Finally, the direct relation between delta power and entropy will be presented in the following picture.

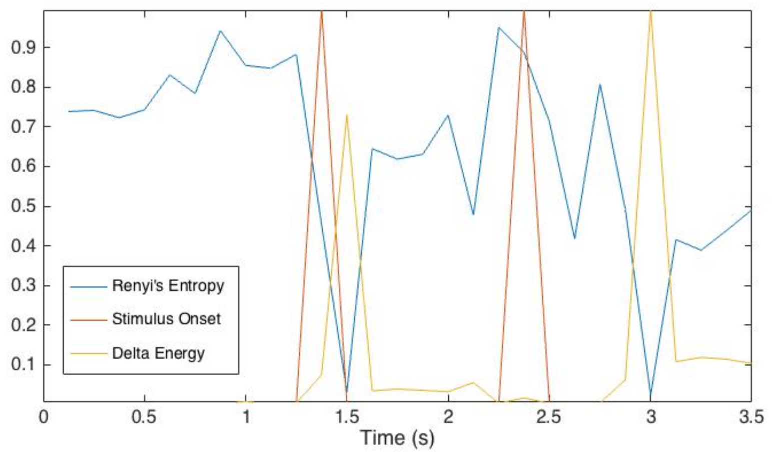


Figure 7.5: Renyi's Entropy and Delta Band Energy

The figure shows a specific time window of 3.5 seconds, it can be clearly seen that the delta peak coincides with the entropy decrease.

It should be noticed that these figures show the entropy and wavelet results preprocessed only with a band-pass filter and later standardization. Also, they correspond to the Fz channel, which showed the best results of all the channels recorded.

In order to prove the improvement on the P300 detection with spatial filtering, the following picture will present the major improvement achieved by performing FastICA and the xDAWN on the signal.

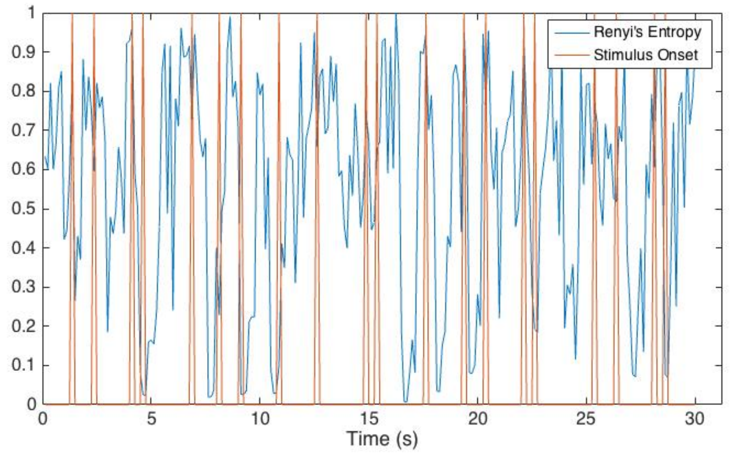


Figure 7.6: Improved Feature Extraction

More minimum values can be seen in this last figure. This represents that the preprocessing steps followed have correctly projected the signal into a subspace where the P300 components are highlighted. This is, precisely, the aim of xDAWN spatial filter.

7.2 Selected Features

Once the potential of wavelet entropy for P300 detection has been proved, this section will present the corresponding extracted features.

The first feature to be analyzed is entropy variance. As it was seen in the previous section figures, the entropy is expected to keep a variance more or less stable expect in those P300 response.

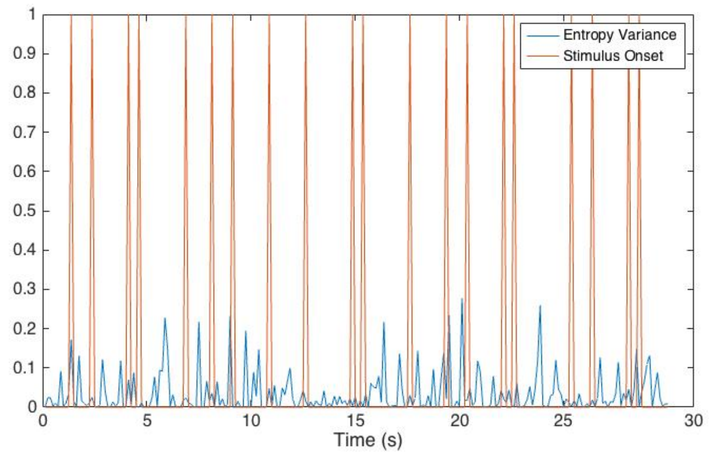


Figure 7.7: Entropy Variance

The second feature is the scaled entropy, which takes advantage of the information provided both from the entropy and the delta energy.

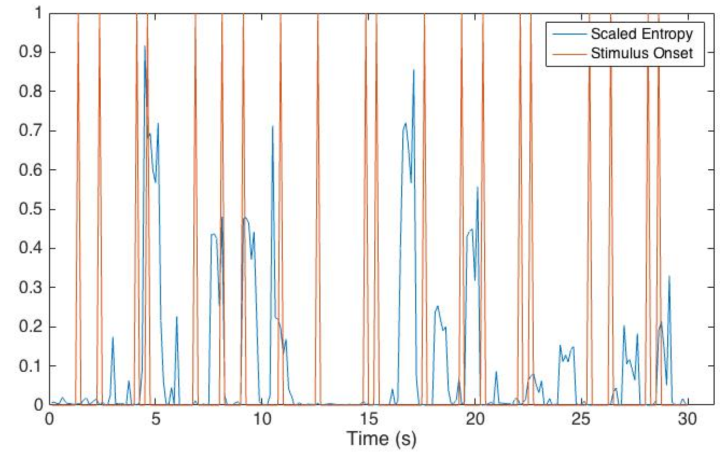


Figure 7.8: Scaled Entropy

Lastly, the inverse entropy will be depicted in the next figure.

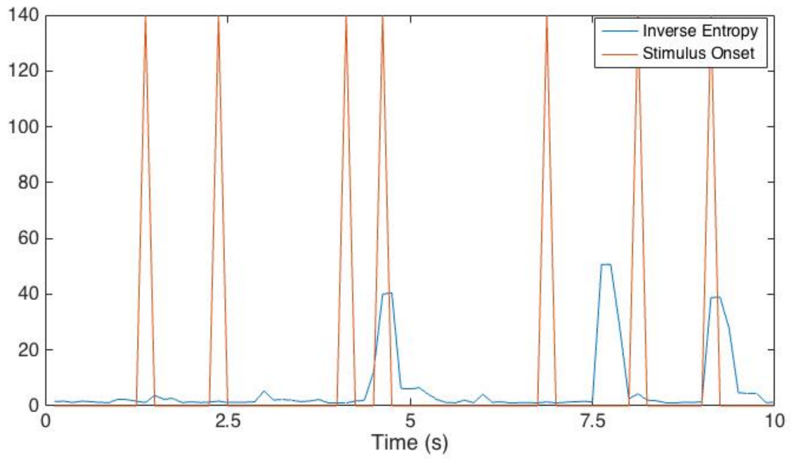


Figure 7.9: Inverse Entropy

By inspecting all the previous figures, some conclusions can be extracted.

First, all of the features are able to show the presence of the P300 wave. Although the entropy variance shows more prominent peaks after stimuli, a very fine selection of a threshold for peak selection must be performed for this feature. Regarding the scaled entropy, it reduces the probability of misclassification since all those intervals where the delta energy is not sufficiently high will not be considered target stimulus. On the other hand, energy can increased due to many other external stimulus and therefore some scaled entropy peaks would lead also to misclassification. Finally, the inverse does show the decrease in its value but it is also affect by the intrinsic variance of the entropy.

Due to all of these reasons, the peak detection was not successful, as it will be seen in the next section.

7.3 Classification Performance

In this section, the final results of the system extracted from the classifier will be presented.

Since several methods were tried out in this thesis, the results here presented will consist of a series of accuracy rates achieved by different combinations of the algorithms.

The following acronyms are used in the table:

- Bandpass → BP
- Standardization → ST
- Entropy Variance → EV
- Scaled Entropy → SE
- Inverse Entropy → IV

Table 7.1: Algorithms used in the project

	Preprocessing	Features Selected	Classification
1	BP + ST	EV + SE + IE	SVM
2	BP + ST + FastICA	EV + SE + IE	SVM
3	BP + ST + FastICA + xDAWN	EV + SE + IE	SVM
4	BP + ST + axDAWN	EV + SE + IE	SVM

And the results for the offline data acquired from BNCI, corresponding to eight subjects, are the following:

Table 7.2: Results from BNCI Data

	Rows Accuracy	Columns Accuracy	Letter Accuracy
1	98.10 %	97.14%	97.14%
2	100%	97.14%	97.14%
3	98.57%	98.57%	98.75%
4	100%	100%	100%

As it can be seen, the results from existing data sets are almost perfect for every single algorithm implemented.

Specially, aXDAWN is the perfect algorithm for the system since it makes no mistake and the accuracy is perfect.

Table 7.3: Results from recorded data

	Rows Accuracy	Columns Accuracy	Letter Accuracy
Two words	90%	100%	90%
Three words	90%	90%	90%
Two words - higher ISI	100%	100%	100%

Results for the recorder data from the P300 Speller designed for the project are also remarkable. These last results correspond to the fourth algorithm, which used the spatial filter axDAWN.
As it was discussed in the literature review, it has been proved that higher ISI increases the performance of the system.

Chapter 8

Discussion

8.1 Wavelet and Entropy

The results presented in the previous chapter showed the theoretically discussed interaction between delta wavelet energy and its entropy. It could be seen that the results assessed the theoretical assumptions.

MRA allowed to adequately separate the EEG signal components. In the specific case of P300 based BCI systems, this is of great importance since the interesting EEG activity takes place in an specific frequency range.

The relationship between the EEG and ERP was identified in the presented results. An ERP corresponds to a transition from a disordered state in the brain signals to an ordered state and so, the entropy consequently decreases [24]. An ordered state provides less information in the context of Information Theory. The EEG resonances upon stimulation can be found only in some of the EEG components. This fact can be easily seen in Figure 7.1 where the energy corresponding to the most relevant decomposition levels is depicted. These oscillations will be also synchronized, which is precisely the reason why an ERP is considered to provoke a more ordered state in the EEG. The spectrum of the EEG will be consequently narrower and the entropy lower.

The family of entropy chosen was the Quadratic Renyi Entropy. This choice was not arbitrary. As discussed in the EEG Processing chapter, the α parameter in Renyi's Entropy allows control over the contributions of the PDF to the entropy. Specifically, higher values make the measurement more sensitive to events which occur more often. This fact could be mistaken when thinking about ERPs since they are low-probability events, specially regarding P300 based BCI systems. However, the wavelet PDF was computed as the ratio of energy between different levels. It has been proved that the delta band has the highest probability of all of the levels. Therefore, in order to highlight the increments of delta energy, α should be selected as a high value, at least higher than the unity, which corresponds to Shannon's entropy. During this project, several values of α were also tested. No difference in the results could be found for values higher than 2.

The preprocessing steps implemented in this thesis highly improved the SNR and managed to project the input signals into a subspace were the P300 components were evident. The better results were provided by the xDAWN filter, in its static or adaptive form. This result was expected since the algorithm is based on seeking a projection of the signal that ressembles the ERPs. Combined with FastICA, the results were further improved.

8.2 Classification Performance

Many algorithms were tested in this thesis and they were combined in a number of ways. All of them were tested with offline data coming from a P300 Speller experiment. Therefore, the accuracy rate will be computed for rows, columns and letters.

All of the systems based on peak detection classification had very poor performance. The reason to this result is, as it has been discussed, the varying delay of time of appearance of the P300 wave. Although the features selected reflect correctly the ERP, this time delay, combined with the small ISI leads to an unacceptable performance for any real-life BCI system.

The introduction of the SVM classifier supposed an increased performance for every one of the algorithms combination. Of course, this improvement is at the expense of higher computational complexity and low speed of execution. However, the accuracy rate is still too low for a BCI system.

The major problem in the classification performance via peak detection was the different rate of accuracy for rows and columns. Only when guessed right together, the letter can be correctly retrieved. Then, a bad combination of rows and column guesses leads to a excessively low rate of success.

Nevertheless, the results showed an accuracy almost perfect for every method thanks to the SVM. The training features for the SVM are those previously explained. As for the training setup time, only 10% of each trial was necessary to train the classifier. The adequate choice of the Gaussian function and the kernel size were the key to the success of the algorithms presented in this thesis. The major drawback of this classifier, as mentioned, is the low speed of execution.

Chapter 9

Conclusion

The aim of this thesis was to develop adaptive filtering techniques to improve the performance of BCI systems. These systems need to be accurate, fast and user adaptive.

The paradigm chosen to develop the BCI system was P300. This ERP is based on low-probability target events that trigger responses in the human brain. Although easy to implement, such a system constitutes a challenge due to the varying nature of the P300 response itself. Not only its amplitude can vary, but also its latency with respect to the stimulus.

An adaptive system seems to be then the appropriate choice for such a varying paradigm. In this thesis several existing algorithms were tested and compared. These algorithms are a combination of existing techniques, which have not been tested together before. Every algorithm had an adaptive scheme, being the spatial filter or the classifier.

Regarding feature extraction, every system was based on the wavelet spectral entropy idea. This feature is a novelty in the BCI field, since there has been little research on its use as a feature for BCI systems. This thesis proved the feature to be effective to detect the P300 wave. Besides, the preprocessing adaptive methods used in the project further improved the wavelet spectral entropy qualities as an P300 estimator.

The developed system was tested in a P300 speller designed for the project. Results showed that the combination of features selected and the SVM classifier represent an almost perfect P300 based BCI system. The only drawback: the execution time.

All in all, this thesis has proved that a combination of adaptive filtering techniques and wavelet spectral entropy has a huge potential for BCI systems. Moreover, the thesis has validated its accuracy in a real BCI-system.

Bibliography

[1] Susan Standring, *Gray's Anatomy*, Churchill Livingstone.

[2] Abou Ella Hassanien and Ahmad Taher Azar, *Brain-Computer Interfaces, Current Trends and Applications*, Springer, 2014.

[3] Bernhard Graimann, Brendan Allison, and Gert. Pfurtscheller, *Brain-computer interfaces : revolutionizing human-computer interaction*, Springer, 2010.

[4] Christoph Guger et al., "Comparison of dry and gel based electrodes for P300 brain–computer interfaces", in: *Frontiers in Neuroscience* 6 (2012).

[5] F Lotte, "A review of classification algorithms for EEG-based brain-computer interfaces", in: *Journal of Neural Engineering* 4 (2007).

[6] Guangyu Bin, Xiaorong Gao, and Yijun Wang, "VEP-Based Brain-Computer Interfaces: Time, Frequency, and Code Modulations", in: *Ieee Computational Intelligence Magazine* (2009).

[7] Fabrizio Beverina, "User adaptive BCIs: SSVEP and P300 based interfaces", in: *Psychnology Journal* 1 (2003).

[8] Gregory McCarthy Scott A. Huettel, "What is odd in the oddball task? Prefrontal cortex is activated by dynamic changes in response strategy", in: *Elsevier Academic Press* (2003).

[9] Heather M. Gray, "P300 as an index of attention to self-relevant stimuli", in: *Journal of Experimental Social Psychology* (2003).

[10] Hansenne N, "The P300 cognitive event-related potential. I. Theoretical and psychobiologic perspectives", in: *Neurophysiol Clin.* (2000).

[11] John Polich, "Updating p300: An integrative theory of P3a and P3b", in: *Clinical Neurophysiology* 117 (2007).

[12] John Polich and Jose R. Criado, "Neuropsychology and neuropharmacology of P3a and P3b", in: *International Journal of Psychophysiology* (2006).

[13] P Sykacek, SJ Roberts, and M Stokes, "Adaptive BCI Based on Variational Bayesian Kalman Filtering: An Empirical Evaluation", in: *Ieee Transactions on Biomedical Engineering* 51 (2004).

[14] Byung Hyung Kim and Sungho Jo, "Real-time motion artifact detection and removal for ambulatory BCI", in: *International Winter Conference on Brain-Computer Interface* (2015).

[15] Yiheng Tu et al., *An automated and fast approach to detect singletrial visual evokedpotentials with application to braincomputer interface*, Elsevier, 2014.

[16] Han Sun and Liqing Zhang, *Subject-Adaptive Real-Time BCI System*, Elsevier, 2014.

[17] Arjon Turnip, Keum-Shik Hong, and Myung-Yung Jeong, "Real-time feature extraction of P300 component using adaptive nonlinear principal component analysis", in: *BioMedical Engineering OnLine* 10 (2011).

[18] Laurent Bougrain, Carolina Saavedra, and Radu Ranta, "Finally, what is the best filter for P300 detection?", in: *TOBI Workshop lll- Tools for Brain-Computer Interaction* (2012).

[19] Hendrik Woehrle, "An Adaptive Spatial Filter for User-Independent Single Trial Detection of Event-Related Potentials", in: *IEEE Transactions on Biomedical Engineering* (2015).

[20] Bertrand Rivet, Antoine Souloumiac, and Virginie Attina, "xDAWN Algorithm to Enhance Evoked Potentials: Application to Brain-Computer Interface", in: *Ieee Transactions on Biomedical Engineering* 16 (2009), pp. 2035–2043.

[21] Mario Michael Krell, Hendrik Wöhrl, and Anett Seeland, "raxDAWN: Circumventing Overfitting of the Adaptive xDAWN", in: *Neurotechnix 2015* (2015), pp. 68–75.

[22] Farwell and Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related-potential", in: *Elsevier Scientific* (1988).

[23] Jessica Lu, William Speier, and Xiao Hu, "The Effects of Stimulus Timing Features on P300 Speller Performance", in: *Clinical Neurophysiology* 124 (2013).

[24] R Quian Quiroga, O A Rosso, and E Başar, "Wavelet entropy: a measure of order in evoked potentials", in: *Electroencephalography and Clinical Neurophysiology* (1999).

[25] Bahram Perseh and Ahmad R. Sharafat, "An Efficient P300-based BCI Using Wavelet Features and IBPSO-based Channel Selection", in: *Journal of Medical Signals and Sensors* 2 (2012).

[26] Virginia R. Hammon Paul S.and de Sa, "Preprocessing and Meta-Classification for Brain-Computer Interfaces", in: *Ieee Transactions on Biomedical Engineering* 54 (2007).

[27] Grzegorz Rutkowski, Krzysztof Patan, and Paweł Leśniak, "Comparison of Time-Frequency Feature Extraction Methods for EEG Signals Classification", in: *Lecture Notes in Computer Science* (2013).

[28] Sebastian Raschka, *Principal Component Analysis in 3 Simple Steps.*

[29] Nikolaos Mitianoudis, "Audio Source Separation Analysis", MA thesis, 2008.

[30] Aapo Hyvärinen and Erkki Oja, "Independent Component Analysis: Algorithms and Applications", in: *Neural Networks Research Centre* (2000).

[31] Benjamin Blankertz, “Optimizing Spatial Filters for Robust EEG Single-Trial Analysis”, in: *IEEE Signal Processing Magazine* (2007).

[32] Ingrid Daubechies, *Ten Lectures on Wavelets*, SIAM.

[33] Claude E Shannon, “The Mathematical Theory of Communication”, in: *J. Clin. Neurophysiol.* (1997).

[34] José Principe, *Information Theoretic Learning - Renyi's Entropy and Kernel's Perspective*, Springer.

[35] PK Sadasivan et al., “Entropies for detection of epilepsy in EEG”, in: *Computer Methods and Programs in Biomedicine* (2005).

[36] Giorgos A. Giannakakis, Nikolaos N. Tsiaparas, and Monika-Filitsa S. Xenikou, “Wavelet Entropy Differentiations of Event Related Potentials in Dyslexia”, in: *8th Ieee International Conference on Bioinformatics and Bioengineering* (2008).

[37] Zachary Cashero, “Comparison of EEG Preprocessing Methods to Improve the Classification of P300 trials”, MA thesis.

Appendix A

MATLAB Code

```
function [ U ] = xDAWN( X, Fs, stimulus )
% X(Nt,Ns) −> Input data, Nt is the no of samples and Ns is the no of
% sensors (channels)
% D(Nt,Ne) −> Toeplitz Matrix, Ne is the no of samples of the ERP
% (tipically 600 ms or 1s)
% D(tk,1) = 1 −> tk is the stimulus onset of the target kth target stimulus
% Stimulus −> start of the stimulus

%% xDAWN Algorithm
[Nt, Ns] = size(X);
ERP_l = 1;
Ne = round(ERP_l*Fs);
col = zeros(1,Nt);
row = zeros(1,Ne);
col(stimulus) = 1;
row(1) = col(1);
D = toeplitz(col, row);

%% QR Decomposition
[Qx, Rx] = qr(X,0);
[Qd, Rd] = qr(D,0);

%% SVD of Qd'*Qx
prod = Qd'*Qx;
[phi,lambda,psi] = svd(prod,0);
%% Largest singular values
s_d = diag(lambda);
var_s = s_d.*(100/sum(s_d));
perc = 0;
no_sing = 0;
while (perc < 85)
    no_sing = no_sing + 1;
    perc = perc + var_s(no_sing);
end
%% Keep at most i eigenvectors
phi = phi(:,1:no_sing);
lambda = lambda(1:no_sing, 1:no_sing);
psi = psi(:,1:no_sing);
```

```
%% Spatial Filters
U = Rx\psi;
A = Rd\phi*lambda;
end
```

Code A.1: xDAWN Filter

```
function [ X_stand ] = zcore_standardization_segments( X, n_subsegments )

l_subsegments = floor(length(X)/n_subsegments);
s = 1;
e = l_subsegments;

for i = 1:n_subsegments
    X_stand(s:e) = X(s:e) - mean(X(s:e));
    X_stand(s:e) = X_stand(s:e)./(std(X(s:e)));
    s = e + 1;
    e = e + l_subsegments;
    if (i == n_subsegments)
        e = length(X);
    end
end

end
```

Code A.2: ZCore Standardization computed in segments

```
function [ X_stand ] = zcore_standardization( X )
X_stand = X - mean(X);
X_stand = X_stand./(std(X));
end
```

Code A.3: ZCore Standardization

```
% Quadratic Renyi Entropy Estimator
% Inputs
%   coefficients : wavelet coefficients
%   c_length: no. of coefficients in each level
%   n_subsegments: no. of subsegments to compute the WT
%   n_channels: no. of channels recorded
%   length_t: time duration of each segment
function [H, E, T, PDF] = renyi_spectral_entropy_2(coefficients, c_length, n_subsegments,
    n_channels, length_t)
n_dec = length(c_length(:,1,1)) - 1;
n_subsegments_h = min(c_length(:,1,1));
E = zeros(n_dec, n_subsegments*n_subsegments_h, n_channels);
T = zeros(n_subsegments*n_subsegments_h, n_channels);
PDF = zeros(n_dec, n_subsegments*n_subsegments_h, n_channels);
H = zeros(n_subsegments*n_subsegments_h, n_channels);
steps = zeros(n_dec, 2);
overlap = zeros(n_dec,2);
t_length = zeros(n_dec,1);
for i = 1:n_dec
    t_length(i) = (length_t/c_length(i,1,1))*1000;
    steps(i,1) = floor(c_length(i,1,1)/n_subsegments_h);
    steps(i,2) = c_length(i,1,1) - (n_subsegments_h - 1)*steps(i,1);
    overlap(i,1) = t_length(1) - steps(i,1)*t_length(i);
    overlap(i,2) = t_length(i) - overlap(i,1);
    overlap(i,1) = overlap(i,1)*100/t_length(i);
    overlap(i,2) = overlap(i,2)*100/t_length(i);
end
s = 1;
e = c_length(1,1,1);
for j = 1:n_channels
    for k = 1:n_subsegments
        s = 1;
        e = c_length(1,k,j);
        for i = 1:n_dec
            if overlap(i,1) ~= 0
                st = s;
                E(i,n_subsegments_h*(k - 1) + 1 ,j) = ...
                        sum( coefficients(st:st+steps(i,1) - 1,k,j).^2 )/steps(i,1) + ...
                        overlap(i,1)*sum( coefficients(st+steps(i,1):st+2*steps(i,1) - 1,k,j)
    .^2 )/(overlap(i,1)*steps(i,1));
                T(n_subsegments_h*(k - 1) + 1, j) = T(n_subsegments_h*(k - 1) + 1, j) +
    ...
                        E(i,n_subsegments_h*(k - 1) + 1 ,j);
                for l = 2:(n_subsegments_h - 1)
                    E(i,n_subsegments_h*(k - 1) + l ,j) = ...
                        overlap(i,2)*sum( coefficients(st:st+steps(i,1) - 1,k,j).^2 )/(overlap(
    i,2)*steps(i,1)) + ...
                        overlap(i,1)*sum( coefficients(st+steps(i,1):st+2*steps(i,1) - 1,k,j)
    .^2 )/(overlap(i,1)*steps(i,1));;
                    T(n_subsegments_h*(k - 1) + l, j) = T(n_subsegments_h*(k - 1) + l, j)
    + ...
                        E(i,n_subsegments_h*(k - 1) + l ,j);
                    st = st+steps(i,1);
                end
                l = n_subsegments_h;
                E(i,n_subsegments_h*(k - 1) + l ,j) = ...
```

```
                    overlap(i,2)*sum( coefficients(st:st+steps(i,2) - 1,k,j).^2 )/(overlap(
    i,2)*steps(i,2));
                T(n_subsegments_h*(k - 1) + l, j) = T(n_subsegments_h*(k - 1) + l, j) + ...
                    E(i,n_subsegments_h*(k - 1) + l ,j);
                s = s + c_length(i,k,j);
                e = e + c_length(i+1,k,j);
            else
                st = s;
                for l = 1:(n_subsegments_h - 1)
                    E(i,n_subsegments_h*(k - 1) + l ,j) = ...
                        sum( coefficients(st:st+steps(i,1) - 1,k,j).^2 )/steps(i,1);
                    T(n_subsegments_h*(k - 1) + l, j) = T(n_subsegments_h*(k - 1) + l, j)
    + ...
                        E(i,n_subsegments_h*(k - 1) + l ,j);
                    st = st+steps(i,1);
                end
                l = n_subsegments_h;
                E(i,n_subsegments_h*(k - 1) + l ,j) = ...
                    sum( coefficients(st:st+steps(i,2) - 1,k,j).^2 )/steps(i,2);
                T(n_subsegments_h*(k - 1) + l, j) = T(n_subsegments_h*(k - 1) + l, j) + ...
                    E(i,n_subsegments_h*(k - 1) + l ,j);
                s = s + c_length(i,k,j);
                e = e + c_length(i+1,k,j);
            end

        end
    end
end
%% PDF as the ratio of the power of the segment to the total power
for j = 1:n_channels
    for k = 1:n_subsegments*n_subsegments_h
        for i = 1:n_dec
            PDF(i,k,j) = E(i,k,j)/T(k,j);
        end
        H(k,j) = -log(sum(PDF(:,k,j).^2));
    end
end
end
```

Code A.4: Quadratic Renyi Estimator

```
% FastICA method based on Gaussian Negentropy
function [ X_out, W ] = fastICA(X)
eps = 1e−3;           % Convergence criteria
maxIters = 1000;      % Max no. of iterations
X = center(X);
X = whiten(X);
% X = X';
[n,m] = size(X); % n:dimensions; m:samples

% Random initial weights
c = n;
W = zeros(n,c);
one = ones(m,1);

% FastICA w/ Gaussian negentropy
for i = 1:c
    w = rand(n,1);
    w = w./sqrt(sum(w.^2));
    k = 0;
    err = inf;
    while (err > eps) && (k < maxIters)

        ws = zeros(n,1);
        k = k + 1;
        wlast = w;
        u = w'*X;
        g = u.*exp(−0.5*u.^2);
        g1 = (1 − u.^2).*exp(−0.5*u.^2);
        w = (1/m)*X*g' − (1/m)*g1*one*w;

        % Decorrelate
        for j = 1:(i−1)
            ws = ws + (w'*W(:,j))*W(:,j);
        end
        w = w − ws;

        % Normalize
        w = w./sqrt(sum(w.^2));

        % Compute the error
        err = 1 − dot(w,wlast);
    end
    W(:,i) = w;

end
X_out = W'*X;

X_out = X_out';
end
```

Code A.5: FastICA

```matlab
% This script evaluates  all  of the  trials  in the BNCI file.  It  uses the
% function 'evaluate_trial_svm.m', which performs the second method
% proposed in the thesis
close  all
clear  all
clc
% Load any BNCI file
load A02.mat
[X, y,  trial ,  y_stim, Fs]   = obtain_data( data );
%% Find real targets for every  trial  to check the  classification  accuracy
real_target   = find_targets( data );
%% Obtain results for each  trial
rows_correct = 0;
cols_correct  = 0;
letter_correct  = 0;

number_of_trials = length(trial);
n_channels = size(X,2);
duration = 30;

% Evaluate each trial
for  i  = 1:number_of_trials

    n_trial  = i;

    % Get the result  for  every  trial  as an array containing the guessed
    % rows and columns
    [  result  ]  = evaluate_trial_svm( X, y, y_stim, trial,  n_trial ,  duration,  n_channels, Fs );

    % A voting process counts how many times a row or a column has appeared
    [rows,  cols ]  = voting_svm(result);

    % Find targets as the row/column with the highest score to check the
    % accuracy of the algorithm
    t_row = find(rows == max(rows));
    t_col = find(cols  == max(cols));
    % Check if the system guessed correctly
    r  = real_target(n_trial ,:) ;
    if  any(t_row == r(1))
        rows_correct = rows_correct + 1;
    end
    if  any(t_col == r(2))
        cols_correct  = cols_correct + 1;
    end
    if  any(t_row == r(1)) && any(t_col == r(2))
        letter_correct  = letter_correct  + 1;
    end
end
```

Code A.6: Test Method 2

```matlab
% This script evaluates all of the trials in the BNCI file. It uses the
% function 'evaluate_trial_svm.m', which performs the third method
% proposed in the thesis
close all
clear all
clc
% Load any BNCI file
load A01.mat
[X, y, trial , y_stim, Fs] = obtain_data( data );
%% Find real targets for every trial to check the classification accuracy
real_target = find_targets( data );
%% Obtain results for each trial
rows_correct = 0;
cols_correct = 0;
letter_correct = 0;

number_of_trials = length(trial);
n_channels = size(X,2);
duration = 30;

% Evaluate each trial
for i = 1:number_of_trials

    n_trial = i;

    % Get the result for every trial as an array containing the guessed
    % rows and columns
    [ result ] = evaluate_trial_xdawn_svm( X, y, y_stim, trial, n_trial, duration, n_channels,
      Fs );

    % A voting process counts how many times a row or a column has appeared
    [rows, cols] = voting_svm(result);

    % Find targets as the row/column with the highest score to check the
    % accuracy of the algorithm
    t_row = find(rows == max(rows));
    t_col = find(cols == max(cols));
    % Check if the system guessed correctly
    r = real_target(n_trial,:) ;
    if any(t_row == r(1))
        rows_correct = rows_correct + 1;
    end
    if any(t_col == r(2))
        cols_correct = cols_correct + 1;
    end
    if any(t_row == r(1)) && any(t_col == r(2))
        letter_correct = letter_correct + 1;
    end
end
```

Code A.7: Test Method 3

```
% This script evaluates all of the trials in the BNCI file. It uses the
% function 'evaluate_trial_svm.m', which performs the fourth method
% proposed in the thesis
close all
clear all
clc
load A01.mat
[ X, y, trial , y_stim, Fs ] = obtain_data( data );
%%
real_target = find_targets( data );
n_channels = size(X,2);
duration = 30;
final = duration*Fs - 1;
samples_run = (final+1)/10;
X_st = X;
%% Initialize aXDAWN algorithm

% Update coefficients
lambda_a = 0.2;
lambda_x = 2^(-15);

Nt = Fs;
d = size(X_st,2);
ERP_l = 1; % Length of the ERP, chosen to be 1 second
e = round(ERP_l*Fs);
f = n_channels;
W_est = randn(d,f)*1e-4;
W = randn(d,f);
A = zeros(e,d);
R11 = lambda_x*eye(d,d);
R12 = zeros(d,d);
R12i = (lambda_x^(-1))*eye(d,d);
X_ERP = zeros(e,d);

rw = zeros(length( trial ),6) ;
cl = zeros(length( trial ),6) ;
for p = 1:3
%% aXDAWN
st = 1;
en = samples_run;

for m = 1:length(trial)

    svm = train_svm(m);

    j = 1;
    k = 1;

    rws = [];
    cls = [];
    for j = 1:10

        x = X_st(st:en,:);
        y_trial = y(st:en ,:) ;
        y_stimtrial = y_stim(st:en,:);

        if j > 1
            x = x*W;
        end
        [istarget, result, res, target] = evaluate_axdawn_svm(x, y_stimtrial, Fs, n_channels
     , svm);

        if istarget

            [rows, cols] = voting_svm(result);
            rws = [rws rows];
            cols = [cls cols];
            in = find(target ~= 0);
            i_st = in(1) - round(0.1*Fs);
            if i_st < 1
                i_st = in(1) + 1;
                i_en = in(1) + round(ERP_l*Fs);
            else
                i_en = i_st + e - 1;
                if i_en > length(x)
                    i_en = length(x);
                    i_st = i_en - e + 1;
                end
            end
            X_ERP = x(i_st:i_en,:);

            A = lambda_a*A + (X_ERP-A)/k;
            k = k + 1;
            R11 = (A')*A;

        end

        R12 = (x')*x;
        for i = 1:f
            if i == 1
                Ri1 = R11;
            end
            if i>1
                Ri1 = (eye(d,d) - (Ri1*(wi)*wi')/(wi'*Ri1*wi))*Ri1;
            end
            wi = W_est(:,i);
            wi = ((wi'*R12*wi)/(wi'*Ri1*wi))*R12i*Ri1*wi;
            wi = wi./sqrt(sum(wi.^2));
            W_est(:,i) = wi;
            W(:,i) = ((sqrt(1/((wi')*R12*wi)))*(wi'));
        end

        st = st + samples_run;
        en = en + samples_run;

    end
end
end
% Check the results
guess_row = zeros(length(trial),1);
m_rw = max(rws,[],2);
guess_col = zeros(length(trial),1);
m_cl = max(cls,[],2);
```

```
for i = 1:length( trial )
    ind = find(rw(i ,:) == m_rw(i));
     if ind == real_target(i,1)
         guess_row(i) = 1;
    end
    ind = find(cl(i ,:) == m_cl(i));
     if ind == real_target(i,1)
         guess_col(i) = 1;
    end
end
sum(guess_row)
sum(guess_col)
```

Code A.8: Test Method 4

```
% This function trains the SVM for the first samples of a given trial
function svm = train_svm(n_trial)
load A01.mat
[ X, y, trial , y_stim, Fs ] = obtain_data( data );
%% Retrieve the trial
duration = 30;
n_channels = 8;
%% Retrieve the trial
 final = (duration*Fs - 1)/10;
st = n_trial - 1;
 if st == 0
     st = 1;
 else
     st = st*duration + 1;
end
nd = st + final;
X = X(st:nd,:);
y = y(st:nd,:) ;
y_stim = y_stim(st:nd,:);
%% Preprocessing
for i = 1:n_channels
    X(:,i) = EEG_preprocessing(X(:,i),Fs,0);
end
%% 1. Standardize data
for i = 1:n_channels
    X_st(:,i) = zcore_standardization(X(:,i));
end
[ X_out, W ] = fastICA(X_st);
n_channels = size(X_out,2);
%% 2. Feature extraction -> Multi-Resolution Analysis (Wavelet)
mother_w = 'db4';
n_levels = 6;

length_t = 1.5;
length_subsegments = length_t*Fs;
n_subsegments = floor(length(X_out)/length_subsegments);

C = [];
L = [];
for i = 1:n_channels
    [C (:,:, i), L (:,:, i)] = multi_resolution_analysis( X_out(:,i), mother_w, n_levels,
     n_subsegments );
end
%% Spectral Entropy
[H, E, T, PDF] = renyi_spectral_entropy_2(C, L , n_subsegments, n_channels, length_t);
for i = 1:n_channels
    H(:,i) = H(:,i)/max(H(:,i));
end
% Normalize the energy
for i = 1:n_channels
    for k = 1:(size(L,1)-1)
        E(k,:, i) = E(k,:,i)/max(E(k,:,i));
    end
end
%% Find targets
target_pos = find(y == 2);
```

```
target_time = (target_pos./Fs)*1000; % Target time in ms
res = (length_t/min(L(:,1,1)))*1000;
target_seg = target_time/res;
target_plot = zeros(size(H,1),1);
target_plot( ceil(target_seg)) = 1;
%% Train SVM
H_var = entropy_variance( H, 1);
EH = abs((H(:,1)-max(H(:,1))).*E(1,:,1)');
hh = H(:,1).^(-1);
svm = fitcsvm([hh, H_var, EH],[target_plot], 'boxconstraint', 5, 'KernelFunction', 'rbf', '
      KernelScale', 20);
[label,score] = predict(svm,[hh,H_var,EH]);
find(label==target_plot);
end
```

Code A.9: Train SVM Classifier

```
% This function evaluates each trial and classifies the data with an svm
% classifier .
% For the first samples of the trial , the svm classifier is trained. Then,
% it computes the feature extraction and later it classifies the data.

function [ result ] = evaluate_trial_svm( X, y, y_stim, trial, n_trial, duration, n_channels,
    Fs )
% % Input
% X: data to be analyzed
% y: target/non−target
% y_stim: stimulus
% trial: starting points of the different trials
% n_trial: no of the trial to be analyzed
% Fs: sampling frequency
% duration: length, in seconds, of the trial
% n_channels: no of recorded channels
%% Train svm for the trial
svm = train_svm(n_trial);
%% Retrieve the trial
 final = duration*Fs − 1;
st = n_trial − 1;
 if st == 0
    st = 1;
else
    st = st*duration + 1;
end
nd = st + final;
X = X(st:nd,:);
y = y(st:nd,:);
y_stim = y_stim(st:nd,:);
%% Preprocessing
for i = 1:n_channels
    X(:,i) = EEG_preprocessing(X(:,i),Fs,0);
end
%% 1. Standardize data
for i = 1:n_channels
    X_st(:,i) = zcore_standardization(X(:,i));
end
[ X_out, W ] = fastICA(X_st);
n_channels = size(X_out,2);
%% 2. Feature extraction −> Multi−Resolution Analysis (Wavelet)
mother_w = 'db4';
n_levels = 6;

length_t = 1.5;
length_subsegments = length_t*Fs;
n_subsegments = floor(length(X_out)/length_subsegments);

C = [];
L = [];
for i = 1:n_channels
    [C (:,:, i), L (:,:, i)] = multi_resolution_analysis( X_out(:,i), mother_w, n_levels,
     n_subsegments );
end
%% Spectral Entropy
[H, E, T, PDF] = renyi_spectral_entropy_2(C, L , n_subsegments, n_channels, length_t);
```

```
for i = 1:n_channels
    H(:,i) = H(:,i)/max(H(:,i));
end
% Normalize the energy
for i = 1:n_channels
    for k = 1:(size(L,1)-1)
        E(k,:,i) = E(k,:,i)/max(E(k,:,i));
    end
end
%% Positions of target stimulus
target_pos = find(y == 2);
target_time = (target_pos./Fs)*1000; % Target time in ms
res = (length_t/min(L(:,1,1)))*1000;
target_seg = target_time/res;
target_plot = zeros(size(H,1),1);
target_plot(ceil(target_seg)) = 1;
%% Resolution obtained with the Wavelet Decomposition
res = (length_t/min(L(:,1,1)))*1000;
%% Start Feature Extraction & Classification for every remaining channel
target = zeros(length(H),1);
for i = 1:n_channels
    ch_t = i;
    %% First feature & Threshold
    EH = abs((H(:,ch_t)-max(H(:,ch_t))).*E(1,:,ch_t)');
    %% Statistical Trials - Channel Fz(1)
    % Entropy variance between consecutive samples
    band = 1;
    H_var = entropy_variance( H, ch_t);
    h = H(:,ch_t).^(-1);
    [label,score] = predict(svm,[h,H_var,EH]);
    target = target + label;
end
step = res*1e-3*Fs;
y_stim_res = y_stim(1:step:length(y_stim));
result = [];
for i = 1:length(y_stim_res)
    if target(i) ~=0
        result = [result; repmat(y_stim_res(i),target(i),1)];
    end
end
end
```

Code A.10: Evaluate Trial with SVM

```
% Function to retrieve all data from the BNCI data sets
function [ X, y, trial , y_stim, Fs ] = obtain_data( data )
Fs = 256;
channels = data.channels;
X_in = data.X;
y_in = data.y;
trial = data.trial;
y_stim_in = data.y_stim;
%%
duration = 30; %30 Seconds per trial
%%
X = [];
y = [];
y_stim = [];
for i = 1:length( trial )
    final = duration*Fs - 1;
    X = [X; X_in(trial(i): trial (i) + final ,:) ];
    y = [y; y_in(trial (i): trial (i) + final ,:) ];
    y_stim = [y_stim; y_stim_in(trial(i):trial(i) + final ,:) ];
end
end
```

Code A.11: Obtain Data from BNCI data sets

```matlab
%% Save Info
close all
clear all

% Change these variables for each trial
no_subject = 'S10';
name = 'Cristina Leza';
% Specify the trial
no_trial = '2wordslong';
% Save the channels
channels = {'Fc3'; 'P0z';'Fz';'Fc2';'Fc4';'P3';'P08';'Cz';'C2';'C4';'Oz';'P07';'Cpz'; ...
    'Cp2';'P4';'Pz'};
filename = [no_subject, '_', no_trial, '.mat'];

%% Record

Fs = 256; % Sampling Frequency

ai = USBAmp(Fs); % Initialize Amplifier

eeg = []; % Store Data

% Receive duration of the trial
duration = 0;
seconds = 0;
fprintf('Receive duration of the trial \n')
try
    duration = (udp_connection('receive'));
catch
end
duration = str2num(duration)
fprintf('Waiting for the interface to begin \n')

mssg = '';
try
    mssg = (udp_connection('receive'));
catch
end

start(ai);

% Discard first five seconds
while ai.SamplesAcquired < Fs*5
end
seconds = seconds + 5;
discard = getdata(ai,Fs);

fprintf('Real data \')

while seconds < duration


    while ai.SamplesAcquired < Fs
    end
    eeg = [eeg; getdata(ai,Fs)];
    seconds = seconds + 1;
    fprintf ('Time: %d \n', seconds)
end

 fprintf ('Recording finished\n')

%% Save data

eegdata = struct('File', filename, 'Name', name, 'SubjectNumber', no_subject,...
    'TrialNumber', no_trial , 'Channels', channels, 'Data', eeg);
save(filename, 'eegdata');
```

Code A.12: P300 Speller Record Script

```
% Function to initialize the amplifier
function [ ai ] = USBAmp(Fs)

ai = analoginput('guadaq',1);
n_channels = 16;
addchannel(ai,1:n_channels);

set(ai,'SampleRate',Fs,'SamplesPerTrigger',inf);
set(ai,'TriggerRepeat',inf,'TriggerType','Immediate');
set(ai,'Mode','Normal','SlaveMode','Off');


end
```

Code A.13: Initialize the amplifier

```
function [ varargout ] = udp_connection( actionStr,varargin )

import java.io.*
import java.net.DatagramSocket
import java.net.DatagramPacket
import java.net.InetAddress

port = 5555;
addr = InetAddress.getByName('localhost');



 if strcmpi(actionStr,'send')
    mssg = varargin{1};
    mssg = int8(mssg);
    try
        packet = DatagramPacket(mssg, length(mssg), addr, port);
        socket = DatagramSocket;
        socket.setReuseAddress(1);
        socket.send(packet);
        socket.close;
    catch sendPacketError
        try
            socket.close;
        catch closeError
            % do nothing.
        end % try
        error('%s.m--Failed to send UDP packet.\nJava error message follows:\n%s',mfilename
    ,sendPacketError.message);
    end % try
varargout{1} = '';

 else



    try
        packetLength = 200;
        socket = DatagramSocket(port);
        socket.setReuseAddress(1);
        packet = DatagramPacket(zeros(1,packetLength,'int8'),packetLength);
        socket.receive(packet);
        socket.close;
        mssg = packet.getData;
        mssg = char((mssg(1:packet.getLength)))' ;
        inetAddress = packet.getAddress;
        sourceHost = char(inetAddress.getHostAddress);
        varargout{1} = mssg;



        catch receiveError

            try
                socket.close;
            catch closeError
```

```
                % do nothing.
            end % try

        end % try

end
```

Code A.14: UDP Connection to communicate interface and recording scripts

```matlab
function sequence = P300Speller(words, intensified, ISI)

cgloadlib
cgopen(1,0,0,2)
cgpencol(0,0,0)
cgalign('l','c')
cgmakesprite(4,800,800,0,0,0) % Original matrix
cgmakesprite(3,800,800,0,0,0) % Start Matrix


 initializeMatrix (4);
 initializeMatrix (3);


% ****************************************
S1 = cgflip;
% Set sprite zero
%
cgsetsprite (0)
%
% Wait until Space is pressed
kd(57) = 0;

while ~kd(57)
    kd = cgkeymap;
end
cgdrawsprite(4,4,0)

% Initialize cell to store the order of each iteration
%
n = length(words);
m = length(words{1});
sequence = cell(n,m);


% Start Recording

udp_connection( 'send','start' );

sumtime = 0;
tic
stop = toc+3;
while toc<stop
end
sumtime = sumtime + stop;
for l = 1:n
    word = words{l};
    for j = 1:m
        letter = word(j);
        order = speller();
        sequence{l,j} = order;
        cgmakesprite(1,800,800,0,0,0) % Original Matrix with word
        cgmakesprite(2,800,800,0,0,0) % Flashing Matrix with word

        text = [word ' - ' letter ];
        initializeMatrix (2,text);
        initializeMatrix (1,text);
        if j == 1
            tic
            stop = toc+2;
            while toc<stop
            end
            sumtime = sumtime + stop;
        end

        kd(1) = 0;
        i = 1;
        index = order(i);
        flash(index, 0);
        %
        while (i <= length(order)) && ~kd(1)
            kd = cgkeymap;
            cgflip (0,0,0)
            flash(index, 1) % Go white
            cgdrawsprite(2,4,0)
            tic
            stop = toc+intensified;
            while toc<stop
            end
            sumtime = sumtime + stop;
            cgflip (0,0,0)
            flash(index, 0); % Go almost black
            cgdrawsprite(2,4,0)
            tic
            stop = toc+ISI;
            while toc<stop
            end
            sumtime = sumtime + stop;
            if i ~= length(order)
                i = i + 1;
                index = order(i);
            else
                i = i + 1;
            end
        end
        cgflip (0,0,0)
        cgdrawsprite(2,4,0)
        tic
        stop = toc+3;
        while toc<stop
        end
        sumtime = sumtime + stop;
    end
end
tic
stop = toc+2;
while toc<stop
end
sumtime = sumtime + stop
cgshut
return
```

Code A.15: P300 Speller

```matlab
function initializeMatrix (Key, varargin)

matrix = {'A', 'B', 'C', 'D', 'E', 'F'; 'G', 'H', 'I', 'J', 'K', 'L'; ...
    'M', 'N', 'O', 'P', 'Q', 'R'; 'S', 'T', 'U', 'V', 'W', 'X'; ...
    'Y', 'Z', '1', '2', '3', '4'; '5', '6', '7', '8', '9', '-'};
coordinates = [-160 -100 -40 20 80 140; 140 80 20 -40 -100 -160];

cgsetsprite(Key)
cgpencol(0.5,0.5,0.5)
cgalign('c','c')
cgfont('Arial',35)
% First Column
cgtext(matrix{1,1},coordinates(1,1),coordinates(2,1))
cgtext(matrix{2,1},coordinates(1,1),coordinates(2,2))
cgtext(matrix{3,1},coordinates(1,1),coordinates(2,3))
cgtext(matrix{4,1},coordinates(1,1),coordinates(2,4))
cgtext(matrix{5,1},coordinates(1,1),coordinates(2,5))
cgtext(matrix{6,1},coordinates(1,1),coordinates(2,6))
% Second Column
cgtext(matrix{1,2},coordinates(1,2),coordinates(2,1))
cgtext(matrix{2,2},coordinates(1,2),coordinates(2,2))
cgtext(matrix{3,2},coordinates(1,2),coordinates(2,3))
cgtext(matrix{4,2},coordinates(1,2),coordinates(2,4))
cgtext(matrix{5,2},coordinates(1,2),coordinates(2,5))
cgtext(matrix{6,2},coordinates(1,2),coordinates(2,6))
% Third Column
cgtext(matrix{1,3},coordinates(1,3),coordinates(2,1))
cgtext(matrix{2,3},coordinates(1,3),coordinates(2,2))
cgtext(matrix{3,3},coordinates(1,3),coordinates(2,3))
cgtext(matrix{4,3},coordinates(1,3),coordinates(2,4))
cgtext(matrix{5,3},coordinates(1,3),coordinates(2,5))
cgtext(matrix{6,3},coordinates(1,3),coordinates(2,6))
% Fourth Column
cgtext(matrix{1,4},coordinates(1,4),coordinates(2,1))
cgtext(matrix{2,4},coordinates(1,4),coordinates(2,2))
cgtext(matrix{3,4},coordinates(1,4),coordinates(2,3))
cgtext(matrix{4,4},coordinates(1,4),coordinates(2,4))
cgtext(matrix{5,4},coordinates(1,4),coordinates(2,5))
cgtext(matrix{6,4},coordinates(1,4),coordinates(2,6))
% Fifth Column
cgtext(matrix{1,5},coordinates(1,5),coordinates(2,1))
cgtext(matrix{2,5},coordinates(1,5),coordinates(2,2))
cgtext(matrix{3,5},coordinates(1,5),coordinates(2,3))
cgtext(matrix{4,5},coordinates(1,5),coordinates(2,4))
cgtext(matrix{5,5},coordinates(1,5),coordinates(2,5))
cgtext(matrix{6,5},coordinates(1,5),coordinates(2,6))
% Sixth Column
cgtext(matrix{1,6},coordinates(1,6),coordinates(2,1))
cgtext(matrix{2,6},coordinates(1,6),coordinates(2,2))
cgtext(matrix{3,6},coordinates(1,6),coordinates(2,3))
cgtext(matrix{4,6},coordinates(1,6),coordinates(2,4))
cgtext(matrix{5,6},coordinates(1,6),coordinates(2,5))
cgtext(matrix{6,6},coordinates(1,6),coordinates(2,6))

if Key == 3
    cgfont('Arial',20)
```

```
      cgpencol(1,1,1)
      cgtext('Press space bar to start',-10,-220)

 elseif Key == 1 || Key == 2
      word = varargin{1};
      cgfont('Arial',30)
      cgpencol(1,1,1)
      cgtext(word,-10,200)
end
 cgsetsprite (0)
 cgflip (0,0,0)
cgdrawsprite(Key,4,0)

return
```

Code A.16: Function to initialize the P300 Speller Matrix

```
function flash(index, stop)

% *************************************
matrix = {'A', 'B', 'C', 'D', 'E', 'F'; 'G', 'H', 'I', 'J', 'K', 'L'; ...
    'M', 'N', 'O', 'P', 'Q', 'R'; 'S', 'T', 'U', 'V', 'W', 'X'; ...
    'Y', 'Z', '1', '2', '3', '4'; '5', '6', '7', '8', '9', '-'};
coordinates = [-160 -100 -40 20 80 140; 140 80 20 -40 -100 -160];

% x spacing, y spacing
% *************************************
cgsetsprite(2)
if stop == 0
    cgpencol(0.5,0.5,0.5)
elseif stop == 1
    cgpencol(1,1,1) % White
end
cgfont('Arial',35)

i = mod(index,6);
if i == 0
    i = 6;
end
if index > 6 % Changing column
    cgtext(matrix{1,i},coordinates(1,i),coordinates(2,1))
    cgtext(matrix{2,i},coordinates(1,i),coordinates(2,2))
    cgtext(matrix{3,i},coordinates(1,i),coordinates(2,3))
    cgtext(matrix{4,i},coordinates(1,i),coordinates(2,4))
    cgtext(matrix{5,i},coordinates(1,i),coordinates(2,5))
    cgtext(matrix{6,i},coordinates(1,i),coordinates(2,6))
else % Changing row
    cgtext(matrix{i,1},coordinates(1,1),coordinates(2,i))
    cgtext(matrix{i,2},coordinates(1,2),coordinates(2,i))
    cgtext(matrix{i,3},coordinates(1,3),coordinates(2,i))
    cgtext(matrix{i,4},coordinates(1,4),coordinates(2,i))
    cgtext(matrix{i,5},coordinates(1,5),coordinates(2,i))
    cgtext(matrix{i,6},coordinates(1,6),coordinates(2,i))
end

cgsetsprite(0)

return
```

Code A.17: Flash of the P300 Speller matrix

```
%% Interface Script
%% Save data
close all
clear all

% Change these variables for each trial
no_subject = 'S10';
name = 'Cristina Leza';
type = '3wordstest';

filename = [no_subject, '_', type,'data', '.mat'];
%% Trial
Fs = 256;
words = {'PHONE'; 'EIGHT'; 'SHOES'};
 intensified  = 0.125;
ISI = 0.125;
int_l = 3; % Interval between letters
int_w = 5; % Interval between words
n_w = size(words,1);
n_l = length(words{1});
duration = ( intensified +ISI)*120*n_l*n_w + (n_l−1)*int_l*n_w + n_w*int_w + 5;
udp_connection( 'send', num2str(duration) );
seq = P300Speller(words, intensified , ISI);
sequence = cell( size (seq,1) , size (seq,2) );
targets = cell( size (seq,1) , size (seq,2) );
% Assign targets
targets{1,1} = [4 9];
targets{1,2} = [2 8];
targets{1,3} = [3 9];
targets{1,4} = [2 9];
targets{1,5} = [5 7];
targets{2,1} = [5 7];
targets{2,2} = [3 8];
targets{2,3} = [1 8];
targets{2,4} = [2 8];
targets{2,5} = [2 10];
targets{3,1} = [1 10];
targets{3,2} = [2 8];
targets{3,3} = [3 9];
targets{3,4} = [5 7];
targets{3,5} = [1 10];
for i = 1:size(seq,1)
    for j = 1:size(seq,2)
        order = [];
        target = [];
        s = seq{i,j};
        for k = 1:120
            order = [order; repmat(s(k), intensified *Fs,1) ];
             istarget = any(eq(s(k), targets{i,j}));
            target = [target; repmat(istarget, intensified *Fs,1) ];
            order = [order; zeros(ISI*Fs,1) ];
            target = [target; zeros(ISI*Fs,1) ];
        end
        sequence{i,j} = order;
        stimulus{i,j} = target;
    end
```

```
end
%% Save data
%% Save data

data = struct('File', filename, 'Name', name, 'SubjectNumber', no_subject, ...
    'TypeTrial', type, 'Sequence', sequence, 'Targets', targets, 'Stimulus', ...
    stimulus);
save(filename, 'data');
```

Code A.18: P300 Matrix with 3 words

```
%% Interface Script
%% Save data
close all
clear all

% Change these variables for each trial
no_subject = 'S10';
name = 'Cristina Leza';
type = '2 wordsfirsttrial ';

filename = [no_subject, '_', type,'data', '.mat'];
%% Trial
Fs = 256;
words = {'JUICE'; 'CLOCK'};
 intensified = 0.125;
ISI = 0.125;
int_l = 3; % Interval between letters
int_w = 5; % Interval between words
n_w = size(words,1);
n_l = length(words{1});
duration = (intensified+ISI)*120*n_l*n_w + (n_l-1)*int_l*n_w + n_w*int_w + 5;
udp_connection( 'send', num2str(duration) );
seq = P300Speller(words, intensified, ISI);
sequence = cell(size(seq,1),size(seq,2));
targets = cell(size(seq,1),size(seq,2));
% Assign targets
targets{1,1} = [4 8];
targets{1,2} = [3 10];
targets{1,3} = [3 8];
targets{1,4} = [3 7];
targets{1,5} = [5 7];
targets{2,1} = [3 7];
targets{2,2} = [6 8];
targets{2,3} = [3 9];
targets{2,4} = [3 7];
targets{2,5} = [5 8];
for i = 1:size(seq,1)
    for j = 1:size(seq,2)
        order = [];
        target = [];
        s = seq{i,j};
        for k = 1:120
            order = [order; repmat(s(k), intensified*Fs,1)];
            istarget = any(eq(s(k), targets{i,j}));
            target = [target; repmat(istarget, intensified*Fs,1)];
            order = [order; zeros(ISI*Fs,1)];
            target = [target; zeros(ISI*Fs,1)];
        end
        sequence{i,j} = order;
        stimulus{i,j} = target;
    end
end
%% Save data

data = struct('File', filename, 'Name', name, 'SubjectNumber', no_subject, ...
    'TypeTtrial', type, 'Sequence', sequence, 'Targets', targets, 'Stimulus', ...
    stimulus);
save(filename, 'data');
```

Code A.19: P300 Speller with 2 words

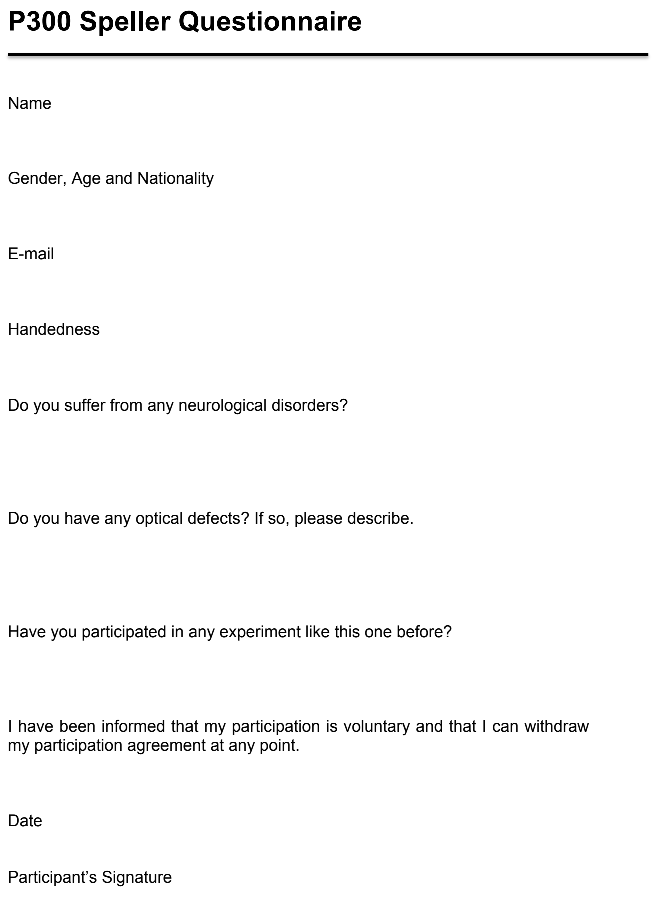
```
%% Interface Script
%% Save data
close all
clear all

% Change these variables for each trial
no_subject = 'S10';
name = 'Cristina Leza';
type = '2wordslong';

filename = [no_subject, '_', type, 'data', '.mat'];
%% Trial
Fs = 256;
words = {'MOUSE'; 'BEACH'};
 intensified  = 0.125;
ISI = 0.250;
int_l = 3; % Interval between letters
int_w = 5; % Interval between words
n_w = size(words,1);
n_l = length(words{1});
duration = (intensified+ISI)*120*n_l*n_w + (n_l-1)*int_l*n_w + n_w*int_w + 5;
udp_connection( 'send', num2str(duration) );
seq = P300Speller(words, intensified, ISI);
sequence = cell(size(seq,1),size(seq,2));
targets = cell(size(seq,1),size(seq,2));
stimulus = cell(size(seq,1),size(seq,2));

% Assign targets
targets{1,1} = [9 1];
targets{1,2} = [3 9];
targets{1,3} = [3 10];
targets{1,4} = [1 10];
targets{1,5} = [5 6];
targets{2,1} = [2 7];
targets{2,2} = [5 7];
targets{2,3} = [1 7];
targets{2,4} = [3 7];
targets{2,5} = [2 8];
for i = 1:size(seq,1)
    for j = 1:size(seq,2)
        order = [];
        target = [];
        s = seq{i,j};
        for k = 1:120
            order = [order; repmat(s(k), intensified*Fs,1)];
            istarget = any(eq(s(k), targets{i,j}));
            target = [target; repmat(istarget, intensified*Fs,1)];
            order = [order; zeros(ISI*Fs,1)];
            target = [target; zeros(ISI*Fs,1)];
        end
        sequence{i,j} = order;
        stimulus{i,j} = target;
    end
end
%% Save data

```

```
data = struct('File', filename, 'Name', name, 'SubjectNumber', no_subject, ...
    'TypeTrial', type, 'Sequence', sequence, 'Targets', targets, 'Stimulus', ...
    stimulus);
save(filename, 'data');
```

Code A.20: P300 Speller with two long ISI words

Appendix B

P300 Speller Pre-Experiment Questionnaire



P300 Speller Questionnaire

Name

Gender, Age and Nationality

E-mail

Handedness

Do you suffer from any neurological disorders?

Do you have any optical defects? If so, please describe.

Have you participated in any experiment like this one before?

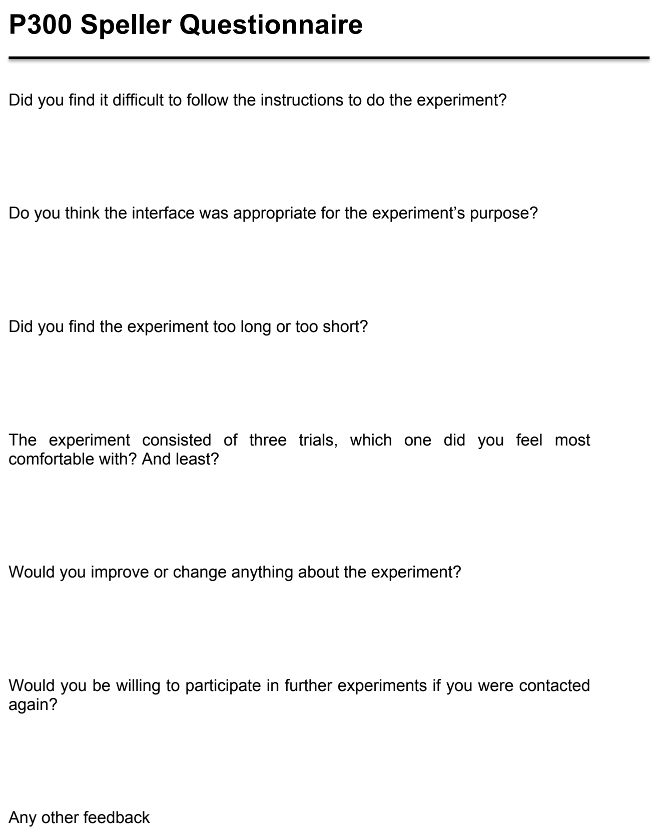
I have been informed that my participation is voluntary and that I can withdraw my participation agreement at any point.

Date

Participant’s Signature

Appendix C

P300 Speller Post-Experiment Questionnaire



P300 Speller Questionnaire

Did you find it difficult to follow the instructions to do the experiment?

Do you think the interface was appropriate for the experiment's purpose?

Did you find the experiment too long or too short?

The experiment consisted of three trials, which one did you feel most comfortable with? And least?

Would you improve or change anything about the experiment?

Would you be willing to participate in further experiments if you were contacted again?

Any other feedback