THE TECHNICAL UNIVERSITY OF DENMARK

BACHELOR THESIS

Motor Imagery-based BCI System for Drone Control

Authors:
Cecilie Saskia LOCHET JAKOBSEN
Jonas HEM ROSSING

Supervisor:
Sadasivan PUTHUSSERYPADY
Co supervisor:
Jakob JAKOBSEN

June 1, 2017

DTU Electrical Engineering
Department of Electrical Engineering

DTU Space
National Space Institute

Motor Imagery-based BCI System for Drone Control

Authors
Cecilie Saskia Lochet Jakobsen, *s144311*
Jonas Hem Rossing, *s140220*

Supervisor
Sadasivan Puthusserypady, Associate Professor, Ph.D.

Co supervisor
Jakob Jakobsen, Researcher, Ph.D.

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

Project period	February 1st 2017 - May 31th 2017
ECTS points	15
Education	Bachelor of Science
Field	Biomedical Engineering (Medicine & Technology)
Class	Open source
Edition	1st
Comments	This thesis is part of the requirements for graduation in the above education at the Technical University of Denmark
Copyright	©Cecilie Saskia Lochet Jakobsen and Jonas Hem Rossing, 2017

Signatures

Cecilie Saskia Lochet Jakobsen — Date

Jonas Hem Rossing — Date

Preface

This bachelor thesis on motor imagery-based brain computer interface for drone control has been conducted as part of the requirements for obtaining a bachelors degree within the field of Biomedical Engineering at the Technical University of Denmark (DTU) and the University of Copenhagen (KU).

The project was initiated in February 2017 and finished in May 2017 at the Department of Electrical Engineering in collaboration with DTU Space and was supervised by main supervisor Sadasivan Puthusserypady[1] and co supervisor Jakob Jakobsen[2].

The total workload of the project represents 15 ECTS points pr. student. The study consists of this final thesis along with all relevant MATLAB ® and Python codes, which are included in the Appendix.

[1] Associate Professor at DTU, Department of Electrical Engineering.
[2] Researcher at DTU Space, National Space Institute.

Acknowledgements

First and foremost, we would like to express our gratitude to our supervisor Sadasivan Puthusserypady for introducing us to the intriguing field of brain computer interfaces and granting us the opportunity of engaging in the exciting combination of BCI and drone control. In addition, we are truly thankful for his support and guidance throughout the writing and development of this project and his capability of communicating complex concepts within the field of signal processing in a comprehensive and distinct manner.

Secondly, we would like to thank our co supervisor Jakob Jakobsen from DTU Space for making drones available for this project and his bachelor student, Mikkel Bugge, for developing an interface between the BCI system and the drone. Furthermore, we would like to express our gratitude towards Per Bækgaard from DTU Compute for providing guidance with regards to the equipment and software used in the acquisition of data during the project.

A special thanks to Martin Simon, Andrei Chiuzbaian and James Hou for the tremendous team work surrounding the data acquisition process and the many insightful discussions regarding the equipment and the signal processing, which took place during this project. Moreover, we would like to thank Ana Palma for her help in understanding common spatial filtering and for sharing the algorithm she developed on the subject. Jacob Petersen should also be acknowledged for providing inspiration for the design of the visual interface used during the data acquisition process in this project.

Finally, we would to express how grateful we are for the moral support and patience exerted by our family and friends during the writing of this thesis.

Abstract

Brain-computer interfaces (BCIs) allows encephalic activity to control an external device or computer. This implies that by recording the signals emitted by the brain, commands can be defined and extracted from the subject to control a variety of devices in a virtual or real world and this technology can be applied to achieve a variety of goals.

Millions of people worldwide are restricted to moving in a wheelchair due to either spinal cord injuries or motor neuron diseases such as amyotrophic lateral sclerosis. Regardless of what disability caused this restriction in movement, the consequences with regard to independence and quality of life can be devastating. A BCI system for controlling a drone may help these patients regaining a certain degree of independence and thus elevate their quality of life. The aim of this project was to design and develop an electroencephalography(EEG)-based BCI system for drone control to provide this opportunity to the described patient group.

To meet this goal, EEG data was acquired from two different subjects. The data set was of 5 different motor imagery tasks, representing the 5 movements the drone would be capable of performing, and the relaxed state of the brain, i.e. idling state, which would correspond to the drone staying in position.

The data was preprocessed with a bandpass filter and 16 different features were proposed to be extracted from the filtered EEG signals of which 13 were used. A feature selection code was developed to determine the optimal combination of these features to be used in the developed system. Furthermore, different classifiers were investigated, which lead to the use of a Support Vector Machine for the classification. Moreover, two live systems were developed to allow for subject training and assessment of the subjects ability to control an aircraft in virtual reality prior to controlling a drone in the real world.

The resulting BCI systems performance was evaluated with several metrics and the maximal classification accuracy obtained during the offline testing of the system was 71.7%.

Resumé

Hjerne-computer interfaces (HCI) gør det muligt at controllere eksterne enheder og computer med hjerne aktivitet. Dette antyder, at det er muligt at optagne signaler udsendt af hjernen kan definere og ekstraher kommandoer fra en forsøgsperson for at kontrollere en række enheder i en virtuel- eller i den virkelige verden og denne teknologi kan anvendes til at opnå en lang række ting.

Millioner af folk verden over er begrænset til at bevæge sig med en kørestol enten på grund af rygmarvsskader eller motor neuron sygdomme, så som amyotrofisk lateral sklerose. Uanset hvilket handicap, der har forårsaget denne bevægelsesrestriktion, så kan konsekvenserne i forhold til uafhængighed og livskvalitet være katastrofale. Et HCI system til kontrol af en drone kan muligvis hjælpe disse patienter med at tilbage vinde en smule af deres frihed og derved øge deres livskvalitet. Målet med dette projekt er at designe og udvikle et electroencephalografi(EEG) baseret HCI system til at kontrollere en drone, for at give denne mulighed til den omtalte patientgruppe.

For at nå dette mål, så blev EEG data erhvervet fra to forskellige forsøgspersoner. Datasættet var af 5 forskellige motor-imaginationsopgaver, der repræsentere de 5 bevægelser dronen vil være i stand til at udføre, og afslappet tilstand, som svarer til at drone står stille i luften.

Data'en blev preprocesseret med et bandpass filter og 16 forskellige karakteristika blev forslået til karakteristika ekstrahering af et filtrerede EEG signal, af hvilke 13 blev brugt. En karakteristika selektionskode blev udviklet til at bestemme den optimale kombination af disse karakteristika, som blev brugt til udviklingen af systemet. Derudover blev forskellige klassifikatorer undersøgt, hvilket ledte til at "Support Vector Machine" blev brugt til klassifikation. Endvidere blev to live systemer udviklet for at akkommodere træning til forsøgspersonerne og for at vurdere deres evner til at kontrollere et fly i en virtuel verden, før de skulle kontrollere en drone i den virkelige verden.

Det resulterende HCI systems ydeevne blev evalueret med flere forskellige metrikker and den maximale klassifikationspræcision opnået under offline test af systemet var $71,7\%$.

Abbreviations

Table 1: Overview of the abbreviations used throughout the thesis

Abbreviation	Explanation
AAR	Adaptive Autoregressive
ANN	Artificial Neural Network
BCI	Brain-Computer Interface
BGN	Bayesian Graphical Network
BP	Band power
BP NN	Back-Propagation Neural Network
BRI	Brain Robot Interface
CNS	Central Nervous system
CSP	Common Spatial Pattern
CWT	Continuous Wavelet Transform
DFT	Discrete Fourier Transform
DOI	Degree Of Imagery
DWT	Discrete Wavelet Transform
EEG	Electroencephalography
e.g.	*exemplie gratia*, for example
EMD	Empirical Mode Decomposition
EMG	Electromyography
ERD	Event-Related Desynchronization
ERS	Event-Related Synchronization
FBCSP	Filter Band Common Spatial Filter
FFT	Fast Fourier Transform
FT	Fourier Transform
HHT	Hilbert-Huang Transform
HMM	Hidden Markov Model
HSA	Hilbert Spectral Analysis
ICA	Independent Component Analysis
i.e	*id est*, that is
IMF	Intrinsic Mode Functions
LDA	Linear Discriminant Analysis
LSL	LabStreamingLayer
MI	Motor Imagery
NN	Neural Network
OVR	One Versus the Rest
PSD	Power Spectral Density
PSE	Power Spectral Entropy
PNS	Peripheral Nervous System
RBF	radial basis function
ROC	Receiver Operating Characteristics
SCI	Spinal Cord Injury
SD	Standard Deviation
SMR	Sensorimotor Rhythms
SNR	Signal-to-Noise Ratio
SSVEP	Steady-State Visually Evoked potentials
VR	Virtual Reality
VRML	Virtual Reality Modelling Language
WT	Wavelet Transform

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Millions of people worldwide suffer from tetraplegia ('tetra' - greek four, '-plegia'- greek plege blow or stroke), implying partial or total paralysis of all four limbs and torso. The most common aetiology of this disability is a spinal cord injury (SCI) in the cervical segment of the spine which, in the case of tetraplegia, leads to impairment of the motor and/or sensory functions, resulting in impaired functions in the upper and lower limbs, trunk and pelvic organs. Another great number of patients suffer from motor neuron diseases (MNDs), including Amyotrophic Lateral Sclerosis (ALS). MND's are a group of progressive neurological disorders destroying motor neurons, controlling essential voluntary muscle activity. Over time, the ability to control voluntary movements can be completely lost and the patients totally paralysed.

As one can imagine, the profound changes in an individuals quality of life after being diagnosed can be completely devastating, as the disability ruins your independence and to a great extend your freedom. Fortunately, as technology evolves, so does the technical means to help this large patient group regain a certain independence with regard to mobility and thereby improving somewhat the quality of life. Thus, the driving force behind this study has been to try to contribute further to the development of this new technologies to the benefit of this vulnerable group of tetraplegics and MND patients.

1.2 Hypothesis

Tetraplegia and MNDs, does not affect the cognitive functions, which means that most patients in these groups still have full cognitive capabilities, which is a prerequisite for the use of brain computer interface technology, being that the only requirement of the user is his or her ability to think. As one of the most prominent challenges amongst people suffering from tetraplegia or late stages of MNDs is the inability to move autonomously, the combination of a BCI system with the drone technology of today could be a sensible and relevant approach to help these patients restore a certain mobility and thereby improve their quality of life. Taking this hybrid BCI system one step further and combining it with VR-technology, eye-tracking technology and a rotating camera on the drone, would give paralysed patients the ability to experience

movement without any constraints or obstacles. This system could then be modified to suit the individual patients needs. For instance, if the drone and the patient were equipped with microphones and speakers, they would be able to interact and communicate with people without leaving their home. This design may imply the possibility for tetraplegic patients to visit their families and friends without ever leaving home.

1.3 Main aims and objectives

The main aim of this study is to provide part of the hybrid system described in the hypothesis, namely the BCI system which allows the user to control the movement of a drone solely with their brain activity.

1.4 Organisation of the thesis

Chapter 2: Clinical background
In this chapter, the relevant terms and processes in the human nervous system are presented and a basic understanding of how brain signals are obtained through electroencephalography is provided.

Chapter 3: BCI systems
A description of the general structure of an EEG-based BCI system is provided in this chapter. Especially MI-based BCI systems are covered, as these are relevant to the project.

Chapter 4 : State-of-the-art
In this chapter, a historical review covering relevant studies in the field of EEG-based BCI systems is presented.

Chapter 5: Materials and methods
A comprehensive description of the equipment and data acquisition process used for obtaining 6-class MI EEG data sets from subjects during the project is presented in this chapter.

Chapter 6: Signal processing
The signal processing steps for creating the MI-based BCI system are presented in this chapter. These include: preprocessing, feature extraction, feature selection and classification.

Chapter 7: Results
In this chapter, the results regarding performance of both the offline and online MI-based BCI system created during the project are presented and described.

Chapter 8: Discussion
A comprehensive discussion regarding the performances of the resulting BCI systems is provided in this chapter.

Chapter 9: Conclusion
Finally, the work related to this study is summarized in a comprehensive final conclusion.

Chapter 2

Clinical Background

This chapter presents the theoretical foundation required for understanding and studying EEG-based BCI systems. The nervous systems physiology and anatomy is described based on [45][52] and a short explanation of the concepts behind electroencephalography concludes the chapter.

2.1 Neuroanatomy and neurophysiology

From feeling, thinking and talking to moving, virtually everything our body does is controlled by the nervous system.

From an anatomical perspective, it consists of the central nervous system (CNS) and the peripheral nervous system (PNS), where the CNS includes the brain (*encephalon*) and spinal cord (*medulla spinalis*) and the PNS consists of all nerves residing outside of CNS. This includes amongst other the 31 pairs of spinal nerves (*nervi spinalis*) and the twelve pairs of cranial nerves (*nervi cranialis*), with the sole exception of the optic nerves (also known as *cranial nerve II*) which can be seen as a part of CNS. Nerves are defined as bundles of axons connecting CNS to sensory receptors, glands and muscles.

Nerve anatomy and physiology

The core structural and functional unit of the nervous system in general, and for the brain in particular, is the nerve cell or *neuron*, sometimes referred to as "brain cell" in layman's terms. Neurons are electrically excitable cells that processes and transmits information by electrochemical signalling. They typically consist of three main parts: the cell body, *soma*, and two different types of cellular projections, termed dendrites and axons respectively. The nucleus is located in the cell body, from which both the axon and multiple dendrites emanates. These two cellular extensions serve as the intercommunication lines between neurons and form *synapses* with one another at interconnection points between one neurons dendrite and another's axon.

When one of the cytoplasm-filled dendrites of a neuron is stimulated by another neurons axon in a synapse, the stimulus is conducted to the cell body of the neuron. As this neuron

may contain multiple dendrites, it can receive many stimuli and when the sum of these stimuli reach a certain threshold, an action potential is generated. An action potential is caused by ion movements through voltage-gated ion channels, which change the intra- and extracellular concentrations of different ions and it consists of a depolarization of the neuron followed closely by a repolarization. It propagates through the nerve cells axon to a synapse connecting it to one or multiple other electrically excitable cell. Through these mechanisms, the brain is able to send out commands, e.g. move right foot forward, through the nervous system and its intricate network of neurons to obtain a muscle reaction.

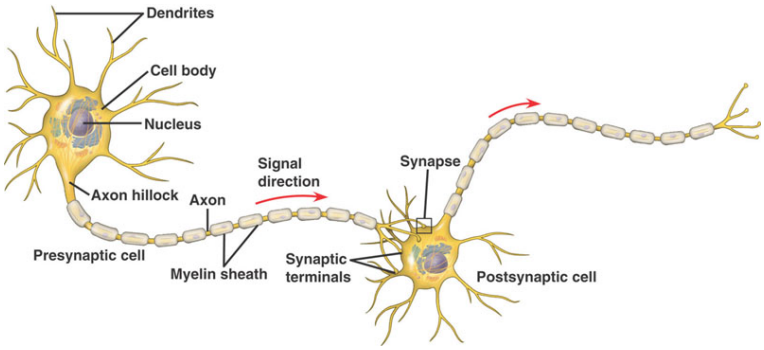


Figure 2.1: Neurons and their anatomy. The neuron consists of dendrites, a cell body (*soma*) containing the nucleus, an axon enveloped in a myelin sheath and the axon terminals. Figure taken from [51].

Apart from the neurons, a number of other cells are present in the nervous system and are termed the *neuroglial cells* as a group. From filling spaces and providing structural framework to produce myelin and carry out phagocytosis, these supportive cells help the nervous system in a variety of ways.

In CNS the neuroglial cells outnumbers neurons greatly and the following four types are present: astrocytes, microglial cells, ependymal cells and oligodendrocytes. The latter of the four types, occurs in rows along the axons of neurons and provide insulating layers of myelin around these in the brain and spinal cord. These *myelin sheaths*, which are interrupted by *nodes of Ranvier*, lowers propagation time in the axons as the action potential propagates from node to node. In PNS, Schwann cells have a similar function as oligodendrocytes in CNS, except they only provide one myelin sheath per cell and an illustration of these can be seen in Fig. 2.1.

Brain anatomy and physiology

The brain (*encephalon*), a wrinkled and soft organ that weighs 1,3-1,45 kg, is located inside the cranial cavity, where the bones of the scull surround and protect it. It contains around 100 billion neurons, which are all heavily interconnected in a cobweb of complexity.

The brain can be subdivided into three major portions: the *cerebrum*, the *cerebellum* and the *brain stem*. Included in the cerebellum, are centers that coordinate muscle movement, while the nerve pathways in the brain stem regulate certain visceral activities and connect various parts of the nervous system. The cerebrum, which is the largest part of the brain, provides higher mental functions, such as reasoning and memory, and contains nerve centers associated with sensory and motor functions. It consists of two large masses called the left and right *cerebral hemispheres*, which are basically mirror images of one another. Each of these cerebral

hemispheres are subdivided into the following lobes, as illustrated in Fig. 2.2, named after the skull bones they underlie:

- The **frontal lobe** forms the anterior part of the cerebral hemispheres and is important in aggression, the sense of smell, mood and motivation. The prefrontal cortex, situated in the anterior region of the lobe, is involved in personality and decision making and the motor cortex, situated in the posterior part of the lobe, is involved in planning, control and execution of voluntary movements.
- The **parietal lobe** is located posterior to the frontal lobe and is the major center for receiving and evaluating most sensory information, except for smell, taste, vision and hearing.
- The **temporal lobe** lies below the parietal and frontal lobes. It plays an important role in memory and receives and evaluates input for hearing and smell.
- The **occipital lobe** forms the posterior portion of each of the cerebral hemispheres and receives and integrates visual input.

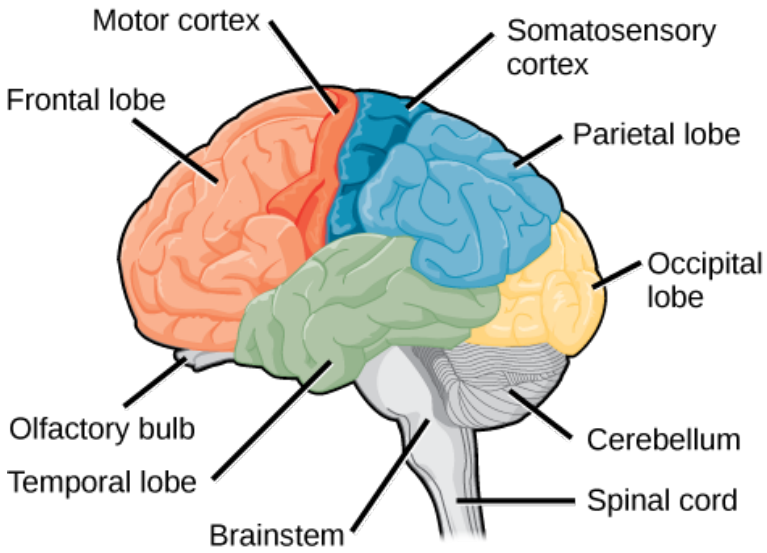


Figure 2.2: The lobes and some of the functional areas of the cerebrum. The figure illustrates the frontal lobe, temporal lobe, parietal lobe and occipital lobe. In addition the locations of the motor cortex and somatosensory cortex are indicated. Furthermore, the figure indicates the position of the cerebellum, brainstem, olfactory bulb and spinal cord. Figure taken from [38].

As this project deals with BCI systems based on motor imagery, which is the imagination of a movement, the most important part of the brain for this project is the motor and somatosensory cortex.

2.2 Electroencephalography

Electroencephalography (EEG) is the recording of electrical activity on the scalp and was first introduced on humans in the 1920s. As the technique is noninvasive and inexpensive, EEG is a very popular method for measuring and utilizing brain activity. It has a good temporal resolution (in the millisecond range), but unfortunately it has a very low spatial resolution,

which is primarily due to the many different layers of tissue between the recording electrodes and the cerebral cortex, as these layers functions as low-pass filters and smudge the signals. Fortunately, multiple preprocessing tools have been developed for improving the low spatial of the EEG signals, e.g. different Laplacian and common spatial pattern filters, as shall be described in Chapter 6.

EEG recordings are often done with so-called EEG-caps on which electrodes are placed in specific positions according to the international 10-20 system, developed to ensure standardized reproducibility of electrode placement. The system is based on measurements of the skull perimeter in both the transverse and median plane, employing locations behind the ears (*processus mastoideus*), on the nose (*nasion*) and at the back of the head (*inion*) as reference electrode placements. The positions of the rest of the electrodes is then determined by 10% and 20% intervals in relation to the distance from nasion to inion (front to back) and from the left mastoid process to the right (left to right). Furthermore, the electrode placements are intuitively named according to the lobe(s) on which the given electrode is placed, as can been in Fig. 2.3.

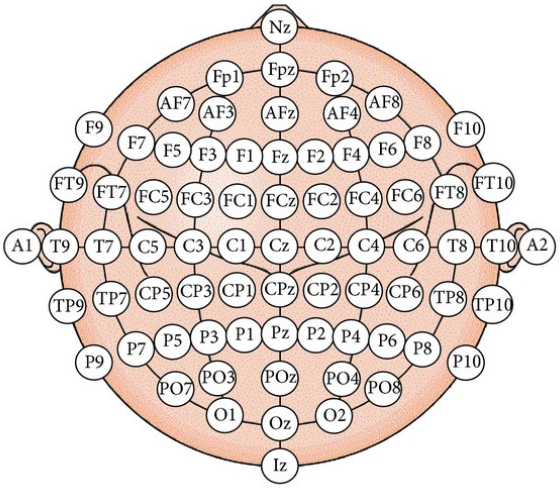


Figure 2.3: The international 10-20 system for electrode placement in electroencephalography, set in place to ensure standardized reproducibility of electrode placement. Figure taken from [21].

As mentioned in the previous section, the brain contains approximately 100 billion neurons and as the electrodes are not sensible enough to measure the activity of one single neuron, the continuous fluctuation of brain activity recorded during an EEG is actually a temporal and spatial summation of activity happening perpendicularly to the given electrode. Each electrode is connected to an amplifier, as the received signals are very small in amplitude and need to be amplified before being displayed for analysis. In addition, the brain signals will often need to be preprocessed before being analysed. A bandpass filter from 0,5 Hz to 100 Hz is often applied on the data, as the brain waves normally oscillates within this frequency range. A notch filter may also be applied to remove noise created by power-line interference occurring at 50 or 60 Hz depending on the country. Furthermore some filtering of the data may be done to eliminate artefacts arising from e.g. muscle activity or eye-movement depending on the primary use of the brain signals in the given case.

In EEG recordings, different oscillatory patterns can be identified and can, to a certain degree, be used to assess a persons mental state and level of conciousness. These patterns are

Table 2.1: The different types of brain wave patterns identified during EEG recordings presented alongside their corresponding frequency range, primary and the conditions under which they are present. Information taken from [20][48][50].

Type	Frequency range	Conditions	Primary location
δ	0,5-3 Hz	REM-sleep, infants	Prefrontal in adults
θ	3-8 Hz	Drowsiness, arousal, idling	Limbic area and frontal regions
μ	8-12 Hz	Absence of movement	Sensorimotor areas
α	8-13 Hz	Resting state with closed eyes	Posterior cortical sites
β	13-35 Hz	Actively concentrated, alert	Parietal and frontal regions
γ	35-100 Hz	Short-memory and motor tasks	Somatosensory cortex

defined by frequency bands/ranges, which varies slightly in the literature, and the different types of brain rhythms and their corresponding frequency bands, frequent locations and conditions are listed in Table 2.1.

The most interesting types of brain waves for this project are the μ and β rhythms, as specific neurophysiological phenomena occur in the frequency bands corresponding to these rhythms during motor imagery. These phenomena are further described in the last section of Chapter 3.

Chapter 3

Brain Computer Interface (BCI)

This chapter seeks to provide a fundamental understanding of brain-computer interfaces. This is sought achieved by giving an introduction to the general structure of such a system, as can be seen in Fig. 3.1, and presenting the different types of BCI systems. In addition, a brief description of some frequently used EEG-based BCI paradigms is provided, with an expanded description of the motor imagery paradigm.

A BCI, sometimes referred brain-machine interface (BMI), translates user intent from encephalic activity into actions in an external device. The concept was first described by Dr. Grey Walter in 1964, but due to limitations in the technology needed for these systems to be effective, the progress of BCI research was essentially at a standstill after its discovery. However, due to developments in technology, the field of BCI research has been growing rapidly in recent years, especially over the past decade.

3.0.1 General structure

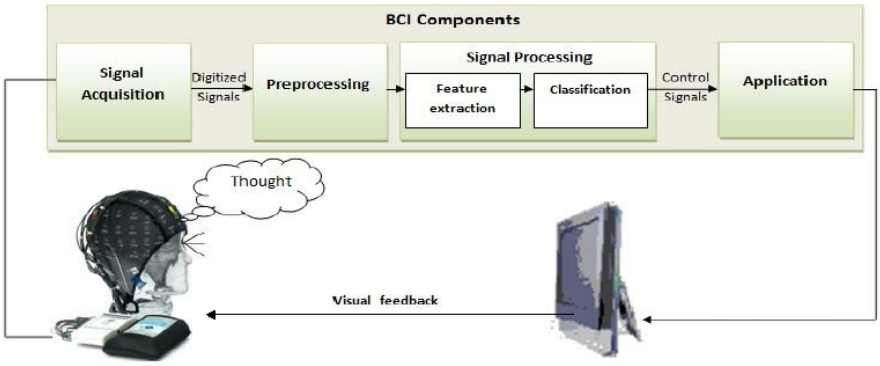


Figure 3.1: The general structure of an EEG-based BCI system. EEG signals are acquired from the subject and digitized, before being preprocessed. The desired features are extracted and classified. The resulting control signal creates an action in the application, which usually provides feedback to the user. Figure taken from [1].

As previously mentioned, a BCI system translates brain activity into actions and the first step to achieve this purpose, is the extraction of a dataset from the encephalic activity. The data can be acquired non-invasively with electroencephalography (EEG) or invasively with either electrocorticography (ECoG) or intracranial electroencephalography (iEEG). After the data is acquired it is digitized by an A/D converter before being fed to the processing unit. Due to different types of noise (*e.g. power line noise*) in the acquired data, it needs to be filtered before the deriving its control signal. Once the data is filtered, specific features from the dataset which reflects the users intent are extracted through feature extraction algorithms. The features are then classified by a classification algorithm and the resulting control signal is then forwarded to the external device or application, where the intended action is executed. Additionally, most BCI system provide some kind of feedback to the user, which is usually visual.

3.0.2 Types of BCI systems

There are multiple ways to characterize a BCI system based on different aspects of the system, whereof some are described in this section based on definitions from [6].

Dependent or independent

One of the important features of a BCI system is that it does not send its commands to the external device through the brains normal output pathways consisting of nerves and muscles. However activity in these pathways might be necessary for generating the needed brain signals for the system to work. If the latter is the case, the concerned BCI system is considered as a dependent system, as opposed to an independent system, where activity in the brains natural output pathways is unnecessary for obtaining the commanding brain signals. Good examples of dependent BCI systems are most of the VEP-based and SSVEP-based BCIs, as the measured brain activity commanding the external device is carried from the subjects eye to the visual cortex in the occipital lobe by the optic nerve or cranial nerve II. Such systems are thereby *dependent* on activity in the optic nerve for generating command signals from the brain and are hence categorized as dependent systems. Most BCI systems are independent and motor imagery-based BCIs fall under this category, as the subject simply imagines movements to evoke the needed brain signals for commanding the external device. However, if one were to use actual movements for provoking the same brain response as with MI, the system would be dependent since muscle activity and thereby activity in the peripheral nerves is used to generate the brain signals extracted for controlling the device.

Synchronous or asynchronous

Synchronous or cue-paced BCIs are cue-based systems which are restricted to predefined time frames. These BCI systems depend on protocols determining the onset, offset and duration of the different operations and are hence limiting, as they do not allow subjects to control them at any desired moment. As an example, the subject is placed in front of a screen and instructed to wait for a cue in the form of a beeping sound before being presented with a task on the screen for him or her to perform to produce the desired EEG signal. The task command then disappears from the screen after a predetermined amount of time, the subject stops the mental task and the system provides feedback about his or her performance after computing a decision based on the EEG signals received. While feedback is given to the subject, the system usually does not process the EEG signals and the signal processing is hence discontinuous. Asynchronous or self-paced BCIs, on the contrary, are always active and the signal processing in these systems is continuous. This type of BCI system is hence able to identify the subjects rest state (*idling*

state), corresponding to the EEG signals arising in periods where the subject has no intent of controlling the system, as well as the EEG signals arising from the predetermined mental tasks used for commands controlling the system. This enables the subjects to control these systems at any moment they desire and thereby offers a much more natural mode of interaction than the synchronous systems. Examples of asynchronous BCI systems are all the systems developed for controlling wheelchairs [28][54][57].

Offline or online
As a fundamental aspect to BCI is its ability to operate in real time, a BCI system with any real application must therefore be an online system to facilitate direct control. Offline systems can be used for examining the different components in EEG signals and to simulate proposed online systems. In offline systems, the effectiveness of different preprocessing, feature extraction and classification methods can be examined and tested while keeping track of potential disruptive artefacts [16].

3.0.3 BCI paradigms

A BCI system can be based on different paradigms, which are chosen according to the objective/goal of the system in question. Some of the prominent EEG-based BCI paradigms are described below, in the order of the most frequently used to the least based on a survey conducted by Han-Jeong Hwang in 2013 [19].

- **Motor imagery**: MI is described as the imagination of kinesthetic movements of different body parts (*e.g. hands, feet, tongue etc.*). Distinct changes in the μ and β rhythms occurs during motor imagery, which can be used for classification of the users intent.
- **Visual P300**: P300 is an event-related potential (ERP) evoked by infrequent, task-dependent stimuli around 300 ms. after each stimulus. Visual P300 is simply a subcategory P300, where the ERP's are evoked by visual stimuli.
- **Steady-state visual evoked potential**: SSVEP is evoked by a repetitive presentation of a flickering visual stimuli with a certain fixed frequency.
- **Non-motor mental imagery**: Nonmotor mental imagery implies all types of mental imagery except motor imagery. This could for example be spacial navigation, mental calculations or internal singing or speech.
- **Auditory**: The auditory paradigm uses brain signals evoked by auditory stimuli. Examples hereof are auditory steady-state response auditory P300.

A BCI system can also be hybrid, which implies that the system is composed of either two different paradigms, as those mentioned above, or one of these paradigms combined with another type of system, such as an electromyography (EMG)-based system or a gaze-tracking system.

An ideal BCI system takes only signals from encephalic activity as input, which implies that the acquired signals are either recorded using electrodes on the scalp, on the surface of the brain or inside the brain of the subject. This suggests that the use of signals evoked from for example eye motion or blinking in a BCI system, transforms it into a hybrid system.

Motor imagery (MI)

As stated in the previous section, BCI systems can be based on different paradigms. In this project the focus is on the MI paradigm, which is the voluntary imagination of movement without actually performing the movement.

Performing MI or actual movements (AM) has been proven to cause changes in the sensorimotor rhythms (SMRs) over the motor and somatosensory cortices of the brain (see Fig. 2.2 for location hereof). For this reason, BCI systems based on the MI paradigm are sometimes referred to as SMR-based systems in the literature.

Two event-related responses can be observed in EEG recordings of a subject performing MI, namely event-related desynchronization (ERD) and event-related synchronization (ERS). The former refers to a blocking response, which implies a diminishing of the amplitude in the brains electric oscillatory response, whilst the ERS refers to an increase in amplitude of the latter. The neurophysiological changes during MI or AM are highly band specific and occur in the μ- and β-bands. During MI or AM, ERD can be observed in the SMR. Correspondingly, ERS can be observed in the SMR after MI or AM has been performed. The ERS/ERD patterns can be observed in both the right and left cerebral hemispheres, but are more prominent in the contralateral side compared to the ipsilateral side with regards to the imagined movement [15]. This suggests that if a subject imagines moving his or her right hand, it will provoke a measurable ERD in the μ-rhythm over the left side of the motor cortex of the subject, which is also true the other way around. In MI-based systems this can be used to distinct between imagined left and right hand movements [27][33][54].

The ERS which occurs after MI or AM is terminated, is often referred to as β *rebound*, which simply implies that the ERS occurs in the β-band after a movement terminates. Depending on which movement is imagined or performed, the frequency at which this beta rebound occurs varies. This beta rebound phenomenon in EEG, has been proven to be useful in detecting and classifying left and right foot movements [15].

Apart from a decrease in magnitude, the active imagination of a movement typically generates brain activity that is spatiotemporally comparable to the activity produced during actual movements [41]. By training individuals to utilize their SMR to provoke an action on a screen, one can minimize this decrease in magnitude and achieve better results in the given MI-based BCI system. This improvement relies on a neurological mechanism termed neuroplasticity, which refers to the brains ability to reorganize itself by forming new neural connections, thereby modulating its electrical oscillatory responses.

Chapter 4

State of the Art

This chapter provides a literary and historical review, covering some of the recent developments in BCI research and the relevant topics for this study. As the latter falls in both the category of BCI systems in medicine and in entertainment, a short review of some of the studies conducted within both fields in recent years is presented in the two first sections of this chapter.

As this project revolves around a MI-based BCI system, an extensive literature review about these systems is provided in the third section of this chapter. This section also contains a table comparing some of the MI-based BCI systems reviewed during the writing of this chapter with regards to algorithms used, accuracy obtained, number of classes etc. Lastly, this section contains an overview of the feature extraction and classification algorithms used in these systems, based on a very recent review article of the progress in EEG-based BCI systems [32].

The fourth section of this chapter is dedicated to describing the research behind the successful MI-based BCI system for drone control developed by LaFleur et al. in 2013 [27] and the research in the control of a helicopter in a virtual 3D environment which served as a base for the development of this system [10] [44] .

As the resulting BCI system in this study is MI-based and asynchronous, a challenge in its design is the long training phases with durations of up to several months, due to the inherent variability of brainwaves from user to user and the need for a subject naïve to BCI to train his or her brainwaves. Therefore the last section of this chapter is dedicated to research based on reducing these long training periods and an example of how the training sessions for the MI-based BCI system for drone control can be established.

4.1 BCI in medicine

The amount of research done regarding electroencephalography(EEG)-based brain-computer interfaces (BCIs) has been increasing rapidly over the past several decades due to the developments in the technology needed for such systems. The technology has primarily been focused on helping patients suffering from severe motor disabilities, e.g. amyotrophic lateral sclerosis (ALS) and spinal cord injuries, interacting and communicating with the world around them as well as providing different applications in neurorehabilitation [12]. The possibility of non-assisted movement in the form of a BCI-controlled wheelchair for the same patient group has also been researched extensively and as early as 2007, the concept was investigated with a tetraplegic patient in an virtual environment (VE) by Robert Leeb et al. [28]. The spinal cord injured subject was able to generate beta oscillation bursts in the EEG by imagining movement of his paralyzed feet, which Robert Leeb and his colleagues utilized in the first demonstration of an asynchronous MI-based BCI system, through which a tetraplegic subject can control his wheelchair within the VE. The concept has in recent years been taken out of VE and into the real world by multiple research groups: In the year 2014, both a P300-based BCI system presented by D. Wee-Kiat et al. [54] and an SSVEP-based BCI system by R. Zhang et al. [57] were introduced. In 2016 R. Zhang and his colleagues improved their system further and combined it with an MI-based BCI system and automated navigation to achieve an improved brain-controlled intelligent wheelchair [58].

4.2 BCI in entertainment

Despite of the primary aims of BCI research being in medicine, the concept has over the past decade attracted attention from other research fields and with a little imagination it is not hard to visualize it spreading to the general population in the form of entertainment and toys in the future. An example hereof has already been developed and described in 2012 by C. Kapeller et al. [23], where the group demonstrated the use of EEG-based BCI for the real-time control of an avatar in the popular game World of Warcraft. C. Kapeller and his colleagues used an SSVEP-based BCI to provide the user with the ability to control a virtual object in a 3-D world. This concept was also applied to flying a virtual helicopter through an MI-based BCI system in 2010 by A.S. Royer et al. [44]. Royer and his group achieved the latter with a conventional 4 class system typically used for control in 2-D combined with intelligent control strategies to obtain fluent and quick navigation in 3-D. Members of the group later expanded the capabilities of the system to allow for full 3-D control of the helicopter in virtual space and their results were published in 2011 by A. Doud et al. [10]. The reductionist control strategy that was employed by Royer and his colleagues in 2010 [44], proved to be a robust control system and was later used for subject training during trials with a MI-based BCI system for drone control developed by K. LaFleur et al. in 2013 [27]. The latter is described in more details in the last two sections of this chapter.

4.3 Motor imagery(MI)-based BCI

Approximately 56% of all EEG-based BCI related articles from 2007 to 2011 were about MI-based BCIs, according to a thorough literature survey from 2013 on EEG-based BCIs by Han-

Jeong et al. [19]. This makes MI the most frequently used paradigm within the field of EEG-based BCI and there is therefore an abundance of scientific articles on the subject, of which a handful are presented here.

According to an article by Juan Tian and Zhaochen Zhang from 2017 [49], the research on MI-based BCI projects using EEG signals was first carried out by Pfurtscheller et al. in the 1990s. The research demonstrated that ERD was produced in the contralateral brain area during unilateral hand movement or motor imagery, while ERS occurred in the ipsilateral brain area. The experiment conducted used electrode position C_3 and C_4 (see Fig. 2.3 for electrode placements) to demonstrate that during MI, the ERD and ERS in the sensorimotor areas of the cerebral cortex occurred in a symmetrical manner with regards to the two brain hemispheres. Further research showed that obvious ERD/ERS phenomenon during MI occurred mainly within the frequencies of μ-rhythm (8-12 Hz) and β-rhythm (13-28 Hz), demonstrating that the phenomenon is very band specific with regards to frequency. The discovery of these EEG characteristics taking place during MI lead to a simple way of classifying left and right hand motor imagery tasks by exploiting the laterality of μ ERD between contralateral and ipsilateral sides occurring during these tasks. By using the estimation of the band power in the μ-band as a feature, multiple research groups have developed MI-based BCI systems with left and right hand motor imagery [13][37].

In a study by Pfurtscheller et al. from 2006 [40] the researchers further investigated the μ-rhythm (de)synchronisation during different MI tasks. The purpose of the study included investigating the distinctiveness between four different MI tasks, namely imagination of left hand, right hand, feet and tongue movements. Furthermore, the inter- and intrasubject variability of event-related EEG (de)synchronization patterns in the mentioned tasks were investigated as well as the different frequency components involved in these patterns.

Nine right-handed subjects participated in the study and the EEG signals were recorded with 60 scalp electrodes placed in accordance to the international 10-20 system (see Fig. 2.3). Using a cue-based MI, trials were made with tasks presented in random order and trials with artefacts were removed by visual inspection. Percentage power decrease and increase in relation to a reference interval was used to quantify the ERD and ERS respectively. Time-frequency maps obtained at the central electrode positions C_3, C_4 and C_z were used for selecting the μ-band rhythms with the most significant band power increase or decrease during the MI tasks. Adaptive autoregressive (AAR) estimates were used as features and a minimum Mahalanobis distance (MDA) was used to classify the four MI tasks.

The results for left and right hand MI confirmed the previously described ERD patterns in the μ- and β-bands. For the foot and tongue MI it was less clear, however a midcentral ERD and a midcentral ERS was found in the majority of the subjects during foot and tongue MI respectively. The most reactive frequency components were between 9 and 14 Hz during all the MI tasks at all central electrode positions. The frequency discrimination between ERD and ERS revealed a mean frequency of the desynchronized components of 10,9 Hz ($\pm 0,9$) and 12,0 Hz ($\pm 1,0$) for the synchronized components. Furthermore, the ERD affects more broad-banded components within the α-band, while the ERS affected narrow-banded components in the upper α- and lower β-bands. Lastly, a great intra- and intersubject variability in the upper μ components reactivity was observed during the different MI tasks.

When motor execution or imagery ceases, beta rebound (ERS in the β-band) occurs within a couple of seconds after movement or imagery termination [3]. In a study conducted in 2013 by Yasunari Hashimoto and Junichi Ushiba [15], it was demonstrated that this phenomenon could

provide high classification accuracy for BCI systems based on left and right foot MI tasks.

Nine healthy subjects with no prior experience in MI were included in the study and the EEG signals were recorded from 31 scalp electrodes placed in accordance to the international 10-20 system. Five cue-based sessions of 40 trials were performed, whereof the first was a training session with weak motor execution (without imagery) and the remaining four were MI. The tasks were presented in random order and surface EMG were monitored to ensure that no motor execution took place under MI.

During offline analysis, all trials where subjects involuntarily contracted muscles were eliminated from the data. An inter-trial variance (ITV) method was employed to calculate the ERD/ERS values and the beta rebound, β ERD and μ ERD were detected automatically with the time-frequency maps and custom-made detection algorithms. The program determined the subject-specific frequency bands and in these the spectral power was calculated. The band power was used as the feature for classification and a traditional linear discriminant analysis (LDA) was used as a classifier, which was trained on a dataset prior to classification. Apart from the offline synchronous system analysis, the study suggested an asynchronous BCI design for classifying left and right foot MI, which was also evaluated.

A significant β rebound was observed in all nine subjects and the peak of β rebound in frequencies between 23 and 28 Hz occurred on average at 3,6 s (SD = 0,8 s) in the post-imagery period. Additionally, in six subjects both the β and μ ERD were significant. In synchronous mode, the BCI system achieved a classification accuracy of 69,3% on average for all subjects, while in asynchronous mode the average classification accuracy was 65,7%. The researches conclude the study with the suggestion that further studies are required to determine the effect of neurofeedback training on classification accuracy.

Prior to the described study, it had been concluded that left and right foot movements produces nearly identical EEG patterns in contrast to left and right hand movements which, as previously mentioned, can be spatially discriminated with EEG. According to the researchers, this is due to the somatotopic organisation of the motor cortex, as the left and right foot representation areas in the brain are closely situated to each other.

4.3.1 Overview of feature extraction and classification algorithms

Early this year, Xiaoqian Mao et al. published an extensive review article on the progress in EEG-based brain robot interaction (BRI) systems, which refers to a subcategory of BCI systems applied for robot control. The paper includes a comprehensive review of the signal processing components of a BCI system within the three main paradigms used in synchronous and asynchronous BRI systems: SSVEP, P300 and MI. The signal processing components included are the preprocessing, feature extraction and classification and an attempt to summarize the two latter of these components with regards to the MI paradigm concludes this section of the chapter.

Feature extraction methods in a MI-based BCI system

The feature extraction is key in the design of a BCI system and a variety of methods have been applied in MI-based BCI systems, whereof some are described here in accordance to the review by Xiaoqian Mao et al. [32].

Fourier-based Transform (FT)

The FT methods are primarily used for power spectrum density analysis (PSDA). To lower the computational power and thereby the delay from thought to action in an online BCI system, one can use the Fast Fourier Transform (FFT) which is a fast computation algorithm for the Discrete Fourier Transform (DFT). Within the MI paradigm, a few studies have attempted to recognize the MI tasks using FFT. In 2014, Hiroyasu et al. [17] utilized the FFT overlap processing for recognition of right or left handed elbow flexion MI. In another study conducted by Jin et al. [22], the FFT was used to analyse the frequency range of μ and β to obtain the energy of the EEG as a feature for classifying left and right MI.

Wavelet Transform (WT)

The WT provide flexible time-frequency resolution and can be used to analyse non-stationary signals, making it a powerful tool in the analysis of EEG signals. It is based on FT, but has the advantage of the ease to choose different mother wavelet functions. In MI-based BCI, the WT can be applied to localize ERD/ERS components in the time domain with great precision. W.-Y. Hsu and Y.N. Sun used weighted wavelet transform features for motor imagery analysis by utilizing the Continous Wavelet Transform (CWT) [18]. In a study by B. Xu and A. Song [55], the Discrete Wavelet Transform (DWT) was used to execute multiresolution decomposition of the EEG signals to show the distribution of the MI signal in time and frequency.

Hilbert-Huang Transform (HHT)

HHT is a recently developed adaptive data analysis method consisting of Empirical Mode Decomposition (EMD) and Hilbert spectral analysis (HSA), with the key part being EMD. The latter can decompose any complicated data set into a finite number of intrinsic mode functions (IMFs). Within the MI paradigm, HHT is an effective way to extract μ and β rhythms. In 2008, Wang et al. analysed three MI tasks using HHT, decomposing the signal with EMD and calculating the Hilbert spectrum based on two IMFs [53]. Lui et al. crafted a new feature based on HHT in 2011, termed the Degree of Imagery (DOI) and demonstrated improvement in the detection and classification of ERD effect using DOI [31].

Independent Component Analysis (ICA)

The ICA method seeks to find a linear representation of non-Gaussian data in a manner that makes the components as (statistically) independent as possible. ICA is mostly used to separate noise from the raw EEG signals in preprocessing, while usually combining other feature extraction algorithms to classify the different tasks when used in feature extraction. In 2006, M. Naeem et al. investigated the separability of four-class MI data using three different ICA and compared them to Common Spatial Patterns (CSP) and two types of EEG deviations [36].

Common Spatial Pattern (CSP)

The CSP yields a set of spatial filters designed to minimize the variance of one class while maximizing it for the other class. It has proven to be superior spatial filtering technique when compared with other spacial filters, but according to Ortner et al., it needs more electrodes than others. The CSP is an especially effective method for MI classification and many improved CSP-based methods have been presented in recent years. Amongst these is the Filter Band CSP (FBCSP) algorithm which classifies single-trial EEG based on chosen features computed from subject-specific temporal-spatial filters. It has been used to classify 4 classes of MI tasks by both Keng Ang et al. and Chin et al.[2][8].

Classification methods in a MI-based BCI system

Due to the nonstationarity of EEG signals, which are especially apparent in the use of EEG-based BCI systems, the stability of the classifier is essential in the discrimination of MI tasks. The most typically used classification methods in BCI are presented here alongside examples of applications in MI-based BCI systems in accordance to the review by Xiaoqian Mao et al. [32].

Linear Discriminant Analysis (LDA)
With the use of hyperplanes, LDA aims to separate the data representing the different classes. The technique is simple and needs very little computational power. In a two-class problem, the class of a feature vector depends on which side of the hyperplane the vector is, while for multiclass BCI systems the "One Versus the Rest" (OVR) strategy is generally used. The main drawback of this classification method is its linearity, which may lead to poor results when used on complex nonlinear EEG data. In 2014 Chen et al. presented the LDA as a novel classifying method for left and right hand MI [7].

Support Vector Machines (SVM)
SVM discriminates classes by constructing a linear optimal hyperplane induced from the maximum margin principle between two classes. The computer algorithm learns by example to assign classes to tasks and two of the major advantages of the SVM classifier are its flexibility and simplicity in usage. A disadvantage of SVM is a longer computational time than other classifiers. In a study conducted by Rathipriya et al. in 2013 the SVM was used to distinguish between two classes, right hand and foot and left hand and foot, respectively [43].

Neural Networks (NN)
Neural Networks are loosely analogous to the human brain in their workings and are remarkably efficient at classifying data. A multitude of neural networks are utilized in BCI systems and amongst these the Multi-layer Perception (MLP) is the most widely used. The latter is a feedforward artificial NN (ANN), in which the Back-propagation network is th most famous model amongst all the feedforward NN. Back-propagation was used by Hamedi et al. in a study conducted in 2014 to classify time-domain features from three different classes [14].

Bayesian Classifiers
The Bayesian classifiers uses the posterior probability, calculated from the Bayesian formulas according to the prior probability of an object. These classifiers are nonlinear and generative and the classifier group mainly includes naïve Bayes classifier, Hidden Markov Model (HMM) and Bayesian Graphical Network (BGN). Bayesian classifiers are not as widely used in BCI applications as the linear classifiers or neural networks, but the naïve Bayes classifier and HMM have been used in BCI systems. The first of these greatly simplifies learning by assuming the features are independent given class and is mainly used in MI. The HMMs are highly efficient for the classification of time series and was used by H-.I. Suk et al. in 2010 for classifying four classes of MI [47].

Nearest Neighbour Classifiers
Nearest Neighbour classifiers are very simple and work by assigning a feature vector to a class with respect to its nearest neighbour, which can be a feature vector or a class prototype. The k nearest neighbour is the most widely used classifier in this group and corresponds to the majority vote of the nearest k neighbours. It has a good performance in MI and obtains higher accuracy rates than many other classifiers. In a study conducted by Diana Eva and Tarniceriu in 2015, the kNN was used to classify left and right hand MI [11].

4.3.2 Comparison of MI-based BCI studies

In Table 4.1, a comparison of some of the studies reviewed during the writing of this thesis, is presented on the next page. It aims to provide an easy-to-read overview of some of the possible combinations of feature extraction algorithms and classifier schemes applied in various, developed MI-based BCI systems.

The resulting classification accuracies are provided for each of the reviewed systems, to demonstrate the outcome of the various choices made during the studies with regard to algorithms used. Furthermore, the number of EEG channels, subjects and classes are presented, as well as the year the study was published and whether the system was evaluated in real-time (*online*) or not (*offline*). The latter is quite relevant when comparing the BCI systems, as an online system will almost always achieve a lower accuracy than an offline system.

Table 4.1: Comparison of some of the different BCI studies reviewed during the writing of chapter 4. The table provides the following information: The year the study was published, the number of channels used for recordings (Ch.), the number of subjects in the study, the number of classes in the system, feature extraction methods (feat. ex.), classifiers and finally, the accuracy of the system. As there is no official way to quantify the latter in asynchronous BCI systems, the given accuracies may not always be comparable to other studies. The abbreviations used in the table with regards to subject number, correspond to the following: PAD stands for Publicly Available Data and RDS stands for Recorded During Study.

Study	Year	Ch.	Subjects	Classes	Feature ex.	Classifier	System type	Accuracy
[49]	2017	3	3 (PAD)	2	DWT	RBF NN	Offline	91,0%
[39]	2016	14	3 (PAD)	2 & 3	DWT (lvl 5)	ANN	Offline	79,4% & 76,1%
[33]	2016	62	13 & 6 (RDS)	4	AR (16^{th} order)	not specified	Online	*many tasks were performed*
[46]	2014	10	2 (RDS)	4	Wavelet-CSP	SVM	Online	79,3% & 89,5%
[15]	2013	31	9 (RDS)	2	Band power	LDA	Offline	69,3 %
[27]	2013	64	5 (RDS)	4	BCI2000's AR	not specified	Online	82,6%
[10]	2011	64	2 (RDS)	6	BCI2000's AR	not specified	Online	87,2%
[29]	2009	3	1 (RDS)	2	DWT (lvl 6)	BP NN	Offline	92,4%
[28]	2007	1	1 (RDS)	2	log BP	threshold	Online	90,0%
[9]	2006	2	2 (RDS)	2	fuzzy NN	LDA	Offline	77,38%

4.4 MI-based BCI system for drone control

As previously mentioned, a MI-based BCI system for drone control was developed in 2013 by K. LaFleur et al. [27] and the group used prior knowledge of 3D control in BCI systems, which was obtained through previous studies demonstrating control of a virtual helicopter with an MI-based BCI system [10] [44], in the development of this system. In the study, 5 subjects successfully controlled a quadcopter with the developed BCI system acquiring up to 90,5% accuracy. Before the subjects were able to control the quadcopter through the developed MI-based BCI system, they underwent an extensive training phase which is described in the next section of this chapter. To the best of our knowledge, this is the first time the ability to control a flying robot through the sole use of thoughts was demonstrated.

The study consisted of three phases; a training and calibration phase, an experimental task phase and an experimental control phase. During the initial calibration phase, the offline analysis toolbox BCI2000 was used to identify the specific electrodes and frequencies that were most differentially active during the performance of a given imagined movement set, e.g. left hand and right hand motor imagery. The software created spectrograms of the R^2 value for each subject, from which the electrodes and frequency bins with the highest correlation value to a given state could be identified and used in the feature extraction, thereby optimizing the classification for each subject and rendering the system subject-specific. The spectral amplitude of the chosen electrodes at the selected frequency components was used as the only feature and was extracted using the BCI2000's online Autoregressive Filter set with a model order of 16 operating on a time window of 160 ms for spectrum calculation. These time-varying spectral components amplitudes were selected and then continuously integrated by BCI2000 to generate control signals that were sent every 30 ms to a Java program which communicated with the drone through WiFi. The data was acquired with a 64-channel EEG cap with electrodes placed according to the international 10-20 system and it was sampled at 1000 Hz. The data was then filtered from dc to 200 Hz by a Neuroscan Synamps 2 amplifier and no spatial filtering was applied before the data was imported into BCI2000.

During the experimental control phase, subjects attended three days of the protocol with 6-15 trials lasting up to 4 minutes each in each daily session. Subjects were regularly reminded of the importance of minimizing movements during experiments, as to reduce the amount of artefacts in the EEG signals and thereby acquiring a higher accuracy rate of classification. Visual monitoring of the subjects during sessions revealed that eye blinking and muscle movements were minimal during all experimental sessions and no artefact removal was included in the system. The MI tasks were assigned to the control directions of the drone in the following manner: Imagining movement of right hand turned the quadcopter right, imagination of the left hand turned it left, imagining both hands caused the quadcopter to rise and deliberately imagining nothing caused it to fall. A constant forward signal was sent to the drone so the linear velocity of the drone in the absence of rotational movement was measured to be $0,69 \pm 0,02ms-1$. This forward velocity attenuated when the quadcopter was turned. The flight environment was set up in a standard gymnasium where two ring-shaped targets made of light-weight foam were suspended approximately 1 m above the ground. The subjects did not have a direct view of the quadcopter, but were provided with visual feedback via a forward facing camera on the hull of the drone. An AR Drone 1.0 quadcopter was chosen for the experiments due to its strong onboard stabilization, wide range of programmable speeds, smooth range of motion and relatively low cost.

The success rate of the study was assessed by three metrics, namely, percent valid correct (PVC), percent total correct (PTC) and percent partial correct (PPC). The subjects weighted average in the three metrics during the experimental task, were 79,2%, 66,3% and 82,6%, respectively.

4.5 BCI training

An important aspect of MI-based BCI that has been discussed and improved extensively over the past years is the aspect of subject training. Due to the inherent variability of brainwave activity between individuals, a user must often undergo months of training before being able to control a given BCI system [26]. During these months, the user learns to consciously modulate his or her brain signals by looking at a continuous feedback based on brain activity in a region of interest and actively changing their EEG signal to make the feedback match a provided detection target. This ability is an inherent consequence of neuroplasticity, which refers to the brains ability to reorganize itself by forming new neural connections throughout life.

When subjects are training for MI-based BCI systems, they are modulating their SMRs according to which movement they are imaging and with what part of the body. This implies that the higher number of actions or classes in the BCI system, the longer the training phase to achieve high accuracies, as the subject needs to train the modulation of SMR with regards to multiple movements. Another consequence is the need for a variety of training programs, as each pair of actions need to be trained individually. The BCI system for drone control designed by K. LaFleur and his colleagues employed an intuitive training sequence with 5 steps before subjects navigated the AR drone in a real-world environment [27]. The first step of this process was a simple 1-D cursor task, which was used by Yuan et al. in 2008 [56]. In this task, a target appeared on either the left or right side of the screen and subjects were instructed to guide a cursor to the target, by performing motor-imagery of left or right hand movement, while avoiding an invisible obstacle located on the opposite side of the target. With a total of three experimental sessions pr. subject with a minimum 9 trials of 3 min., this task was repeated until the subjects achieved a score of 80% or above in four consecutive 3 min. trials, or until this score was achieved when performance was averaged across 10 or more consecutive trials. In the following task, a second dimension was trained independently from the first with another simple cursor task. The same concept of an appearing target and an invisible obstacle to avoid is applied, but this time the target appears on either the top or bottom of the monitor. The subjects were instructed to imagining curling or squezing both hands to move the cursor up and to move it down through the use of a volitional rest. Rules for progressing to the next training phase were the same as for task 1. The third phase combined the command signals from the two previous phases together in a 2-D cursor task with targets on the top, bottom, left and right sides of the monitor. When subjects could either select 70% or more valid targets in four consecutive 2-D trials or an average of 70% or more of valid targets over ten consecutive 2-D trials, they progressed to the next step. The fourth step of the training sequence is, as mentioned earlier, controlling a virtual helicopter created by Royer et al. [44], which is done over three sessions for a total average of about 1 h of flight time. The last step before navigating the drone in the real world, is an enhanced virtual simulation of the AR drone which mimics the behaviour of the actual drone in a real physical environment better than the virtual helicopter simulation. After one to three sessions with the virtual AR drone, the subjects started training with the real AR drone and those who prior to the study had no BCI experience, spent an average of 5 h and 20 min of virtual training over an average period of three months.

According to a publication from 2015 by N. Kosmyna et al. [26], this training is called operant conditioning (OC) and all of the first BCI systems relied on this form of training. The study by N. Kosmyna et al. [26] mentions the main factors involved in the feed-back part of BCI training and the group presents machine learning and the increasingly powerful computers as important factors in the solution to these long training periods. They justify the latter by stating that these developments allowed advanced signal-processing algorithms, which can be trained to separate noise from EEG-signals and that computers can now be taught how to better recognize EEG signals. A year prior to the aforementioned publication, N. Kosmyna et al. presented a bidirectional feedback system for a MI-based BCI system for drone control [25]. By basing the BCI architecture on co-learning between user and system, the group achieved an operational BCI within minutes, a spectacular improvement from the hours of virtual training spent over an average period of 3 months during the training phase for the drone control BCI system developed by K. LaFleur et al. [27].

Chapter 5

Materials and methods

This chapter presents the procedure surrounding the experimental work realized during this project. First, the equipment and software used for the data acquisition are described, with an introductory discussion of how and why the concerned equipment and software were chosen. The experimental paradigm is then presented, including the setup, recording procedure and visual interface used for the data acquisition of a 6-class MI EEG data set for classifier training. Next, a presentation of the two online systems employed during the project is provided. These systems include a training system in which feedback is presented, in the form of words describing the given imagined movement, to the subject in real-time and a flight simulation program in which the control signals from the subject move a plane around in a virtual reality.

5.1 Equipment and software

During the first couple of weeks of the project, a number of problems regarding the provided equipment were encountered. As the developed BCI system in this project needs to be portable in order to let a tetraplegic patient control a drone from his wheelchair, the chosen EEG system must be wireless and somewhat easily transportable. The g.tech system made available by DTU Acoustics in building 354 was therefore not an option.

An Emotiv Epoc+ neuroheadset was borrowed from DTU compute, as the data transfer is done via bluetooth making the system both wireless and transportable. Upon receiving the borrowed equipment, it was found to be in poor condition with corrosion and broken locking mechanisms on most of the electrodes. Furthermore, the electrode placement on the neuroheadset was not optimal for recording SMR's as the electrodes C_3, C_4 and C_z, which are placed over the sensorimotor cortex, were not available. A meeting was arranged with the owner of the system, Per Bækgaard.

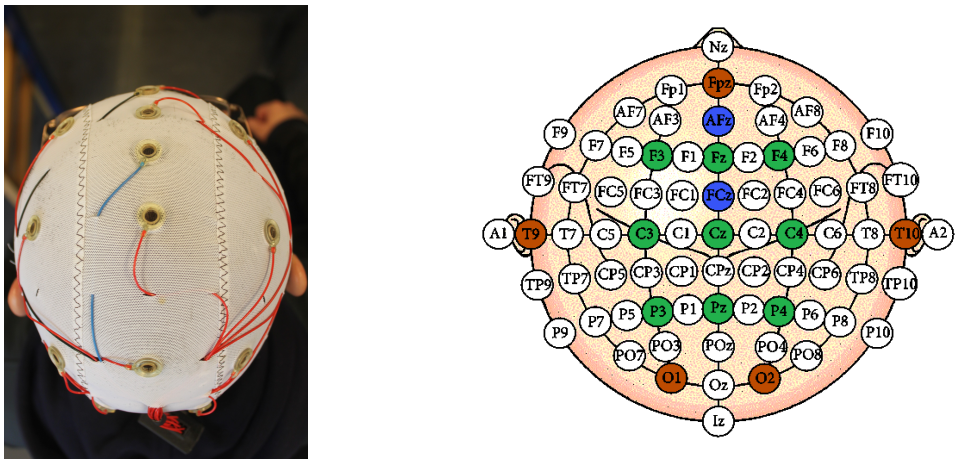
After a productive meeting with Per Bækgaard from DTU Compute, the low quality and fragility of the system became clear; the electrode mounting design and even the design of the electrodes themselves was poorly done and no amount of care or caution could prevent them from breaking after a given time in use. Per Bækgaard presented a "home made" combination of the bluetooth transmitter circuit from an Emotiv neuroheadset and an EEG cap from Easycap with electrodes of significantly better quality than the neuroheadset from Emotiv. Information about Emotivs software was also discussed during the meeting and as the software is user spe-

cific, one needs an ID'd login with Emotiv to acquire the majority of it, which Per could not provide. He instead provided an alternative: A decryption code written in Python with data acquisition possibilities used by DTU Compute to acquire raw EEG data from Emotiv systems.

5.1.1 Equipment

The borrowed EEG cap from Easycap was equipped with 16 electrodes whereof two were reference electrodes. The electrode configuration can be seen in Fig. 5.1. As the electrodes were AgCl electrodes, a highly-conductive saline gel was used to promote the signal conductance from the subjects scalp to the electrode.

The two references on the EEG cap corresponds to the ones used in Emotiv neuroheadsets, where one reference is referred to as the *common mode sense* (CMS) electrode and the other as the *driven right leg* (DRL) electrode. These systems use absolute referencing against the CMS, while the DRL serves as a feedback noise cancellation system. Together, these two electrodes form a feedback loop.



(a) Picture of the Easycap used throughout the project. The electrodes wired with blue and black wires correspond to the two reference electrodes, while the electrodes wired with red wire correspond to the measuring electrodes or channels.

(b) Electrode placement on the EEG cap from Easycap on the international 10-20 system. Blue corresponds to the references, green to the electrodes used by the BCI system and brown to the electrodes available on the cap, but unused by the system.

Figure 5.1: Electrode placement on the EEG cap from Easycap.

Due to the fact that the bluetooth transmitter from the Emotiv system has been connected with the Easycap by DTU Compute, the placement of each of the two types of referencing electrodes could not be determined by the group behind this project. There could therefore be no investigation into the effects of changing the referencing method with regards to the developed BCI system.

Table 5.1: Overview of the Equipment used during the development of the BCI system.

Item	Description
Easycap	EEG-cap with 16 electrodes including two references
Emotiv transmitter	The Emotiv neuroheadsets inner circuit for transmission of EEG data through bluetooth
Emotiv dongle	Receiver dongle connecting the bluetooth transmitter with the computer
Saline gel	Highly conductive saline gel for improving signal conduction from scalp to electrode (Signa gel)
Two PCs	One used for recording the data and the other for displaying the visual interface

5.1.2 Software

The main program used during the development of the BCI system was Matlab. This mathematical programming language was used for preprocessing, feature extraction, feature selection, classification, visual interfaces, feedback mechanisms and simulations.

The data acquisition was performed using Python 2.7, as the available decryption software provided by DTU Compute was written in Python code.

The Emotiv Xavier Control Panel, provided by the Emotiv software webpage, was utilized to monitor the connection between the PC acquiring data and the custom made EEG cap from Easycap. Additionally, the software was used to verify whether the impedances between the electrodes and the scalp were sufficient. As this software was based on the electrode placement on the Emotiv neuroheadset, a translation of the corresponding electrode placement on the EEG cap from Easycap had to be performed in order to label the channels correctly prior to processing the data. The latter took place in Matlab.

One of the challenges in this project, was to transfer the EEG data recorded in Python to MATLAB for processing in real-time. For this, Lab Streaming Layer (LSL) was employed, which is an open source system available on Github and developed for research purposes. LSL can handle real-time communication between computers and a wide range of different software. However, in this project it was only used to transfer the recorded data from Python into Matlab for signal processing.

Table 5.2: Overview of the software used during the development of the BCI system

Program	Description
MATLAB R2017a (9.2)	Mathematical programming language
Python 2.7 (spider)	Programming language
Lab Streaming Layer	Open source C++ based streaming software
Windows 10	Operating system used for recording and processing
Emotiv Xavier Control Panel	Control panel for Emotiv systems

To ensure swift communication between computers, UDP (*User Datagram Protocol*), which is a transport layer protocol defined for use with the IP network layer protocol, was used.

5.2 Data acquisition: 6-class MI EEG data set

Multiple 6-class MI EEG data sets were obtained from two healthy subjects in their 20's, one male and one female, which for future reference will be denoted as subject 1 and subject 2, respectively. It should be noted, that subject 1 suffers from Tourette Syndrome, which is not optimal for BCI systems. Both of the subjects had corrected to normal vision and were naïve to MI prior to the data acquisition process. The low number of subjects in this study is a result of a combination of a lack of time and unstable equipment. The latter was realized during test trials of the equipment as a new issue arose every time it was used.

The objective behind the acquisition of 6-class MI EEG data sets was to utilize the data for training the chosen classifier for the BCI system. It was therefore essential that the recordings were as noise free as possible. For this reason, it was decided to acquire more data sets than needed to train the classifier, to ensure the acquisition of enough high quality data sets.

To train the classifier, a sufficient amount of trials for each class is needed. However, it is important not to feed so much data to the classifier as this may result in overfitting. It was decided to try using two to three data sets, corresponding to 120 and 180 trials, respectively, as this amount of data appeared reasonable when comparing to the literature. Furthermore, this manner provided the ability to compare classification accuracy in relation to the amount of training data and thereby estimate the optimal amount.

The chosen MI tasks described in Table 5.3 were selected based on the MI movement described as identifiable in the articles reviewed during the writing of chapter 4, i.e. the imagined movements that are generally used. Originally, dorsal flexion of the right foot was assigned to class 4 and dorsal flexion of the left foot was assigned to class 5 and two data sets of 5 sessions were produced with these classes. As this type of motor imagery can not easily be spatially discriminated between, as is the case for the left and right hand, it proved difficult to classify these to imagined movements with the selected features. Furthermore, it is significant that the time interval from performed MI task to execution of the corresponding control command is as short as possible in the live system. Due to the latter, the method presented by Hashimoto et al. in 2013 [15], where the imagined movement is identified via the β rebound occurring approximately 3,6 s after termination of the MI task, could not be applied in the development of the BCI system in this project. The movement used as class 4 and 5, were therefore changed to dorsal flexion of both feet and movement of the tongue, which have proved efficient tasks for motor imagery in multiple studies [10][27][39][40].

Table 5.3: Overview of the different motor imagery tasks and their assigned class number. The tabel includes a description of what each of the movement entails and which command the given MI task corresponds to in the developed BCI system.

Class nr. & MI task	Description
1. *Clench right hand*	In this task, the subject imagines clenching his/her right hand with a constant but mild force. This movement will correspond to the drone turning right in the online system.
2. *Clench left hand*	In this task, the same movement as described above is imagined with the left hand of the subject. This gesture corresponds to the drone turning left.
3. *Clench both hands*	The subject clenches both her hands during this task, with the same force as described for the right and left hands. The MI task will correspond to the drone moving forward.
4. *Dorsal flexion of both feet*	During this task, the subject imagined flexing both feet upwards in a dorsal flexion. This MI task will correspond to the drone moving downwards.
5. *Tongue movement*	In this task, the subject imagines moving the tongue either upwards towards the palate or backwards towards the throat. This imagined movement will correspond to elevating the drone.
6. *Relaxation state*	In this task, the subject deliberately focused on nothing and thereby sat in a rested state with relaxed but open eyes. This state, termed the idling state, was recorded as a class to ensure the systems capability of separating the idling state from the MI tasks controlling the drone. The task will correspond to the drone remaining stationary in the air at the given position.

5.2.1 Experimental setup

All of the recordings accomplished during this project were carried out in building 349 of the Technical University of Denmark.

The subject was placed in a comfortable chair in front of a computer screen with his/her arms laying flat on the table in front of him/her. The feet of the subject were placed flat on the ground thus creating a perpendicular angle between shin and thigh.

The computer placed in front of the subject displayed the visual interface during data acquisition, whilst a computer behind the subject acquired the EEG data. The second computer was placed as close to the bluetooth transmitter on the EEG cap as possible, as the bluetooth range had proven poorer than originally expected the dongle needed to be in close proximity to not lose connection. This was also the justification behind the need for two computers. However, this dilemma was resolved through the acquisition of a 3 m long male-female USB extension cord, which was connected to the computer and the receiver dongle. The latter was attached

with the Emotiv transmitter hanging out from the back of the EEG cap with tape, to ensure constant connection between the electrodes on the EEG cap and the recording computer. One would reason that this arrangement rendered the system non-portable, but in reality the subject had more freedom of movement than previously as the USB cables reach was much longer than the relatively fragile bluetooth connection.

(a) **(b)**

Figure 5.2: Experimental setup during data acquisition after optimization, i.e. one computer for both recording the EEG signals and displaying the visual interfaces. The USB extension cord purchased to enable the optimization can be seen in figure (b), where the subject is interacting with the online training system. On this image the bluetooth transmission box is also visible and is clearly connected to the computer via the USB cable.

5.2.2 Recording procedure

As it was decided to use 120 to 180 trials from each of the six classes to train classifiers, the recording procedure was designed on the basis of reaching this desired amount.

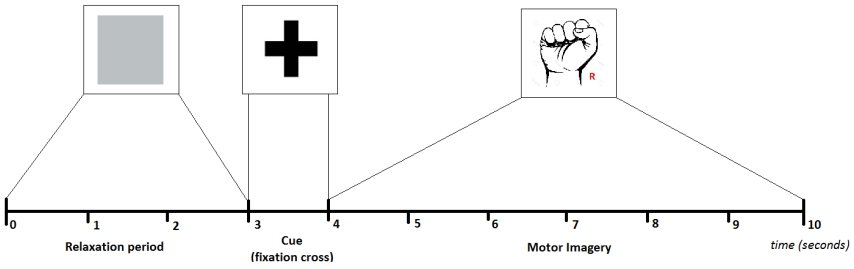


Figure 5.3: Timeline corresponding to one trial during the recording of the 6-class MI EEG data sets. In the beginning of each trial the subject was presented with a gray square for 3 sec, representing the relaxation period. Next, a cue in the form of a fixation cross told the subject to prepare for the MI task. The latter is shown for 1 sec after which one of the MI tasks is presented to the subject for 6 sec.

Each session produced during one run of the visual interface corresponded to 12 trials, two of each class. The timeline corresponding to one trial is visualized in Fig. 5.3. After 5 sessions, a data set of 60 trials, ten for each class, was obtained. For future reference, the definition of a data set is thereby 60 trials throughout the remainder of this thesis.

Enough small breaks were held between sessions to prevent fatigue in the subjects and a longer break was provided after completion of a full data set to the same end goal.

The subject was asked to minimize any activity that could cause noise in the EEG signal, i.e. involuntary muscular activity and blinking. However, if the subject needed to blink during the data recording, he/she was instructed to do so during the relaxation breaks between the MI tasks, as the data recorded during these periods would be discarded before the data sets were fed to a classifier for training. Optimally, the subject should be under observation during the data acquisition to note whether the subject moved or blinked at inappropriate times, so the noise filled trials could be removed before using the data. Due to lack of manpower, this was not possible, but the two subjects reported when he/she moved or blinked to much during a session so a new one could be recorded to replace it.

5.2.3 Visual interface

The visual interface used during the recording of two 6-class MI EEG data sets was developed in Matlab. It was designed to run one session of trials every time it was initiated and thereby displayed the array of figures, shown in Fig. 5.4, twelve times with the various MI tasks. A short relaxation period of 5 s preceded each session when initiating the code to allow the subject to calm down and focus before any measurements were taken.

A function was created to generate random sequences of 12 digits ranging from one to six, corresponding to the 6 different MI tasks. Each task was repeated twice in each sequence to achieve an equal amount of trials for each of the six classes during each session. A new sequence of digits was generated before every session, so the subject could not prepare for a given task and thereby could not perform motor imagination prior to the MI task being displayed, i.e. during relaxation and focus periods.

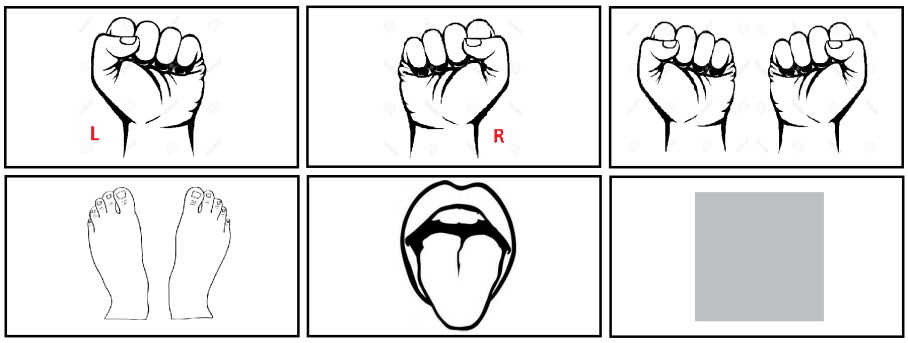


Figure 5.4: The six figures corresponding to the six different MI tasks, presented by the visual interface during data acquisition.

The setup of the visual interface was inspired by a fellow BCI-enthusiastic DTU student, Jacob Petersen.

5.3 Online systems

To asses whether the developed BCI system possessed the ability to classify the MI tasks with sufficiently high accuracy, before using it to control a drone in the real world, some real-time programs were developed. These programs also served as training systems for the subjects, as to improve their capability of performing MI.

5.3.1 Real-time training system

As discussed in Chapter 4, training is an essential part of MI-based BCI systems, as it is necessary for users to modulate their SMRs to enhance the accuracy performance of a given BCI system. Therefore a simple training system was developed, to improve the subjects ability to perform MI. The idea behind the system was to provide the users with live feedback during MI performance, enabling them to evaluate their own ability to perform the different MI tasks. Feedback consisted of a string of text describing the classified imagined movement, which could be presented at a maximum pace of one movement per half a second. As this pace could seem overwhelming to subjects, the training system was programmed to be adjustable with regards to pace and could thereby display feedback at a slower rate if deemed necessary.

The experimental setup was similar to the one used during the acquisition of 6-class MI data sets. A control computer was located behind the subject and would receive and analyse the incoming EEG signals, while another was operating as the user interface as seen in Fig. 5.5. The two computers communicated through the previously described UDP to display the imagined movements classified by the control computer, in the visual interface on the other computer in real-time.

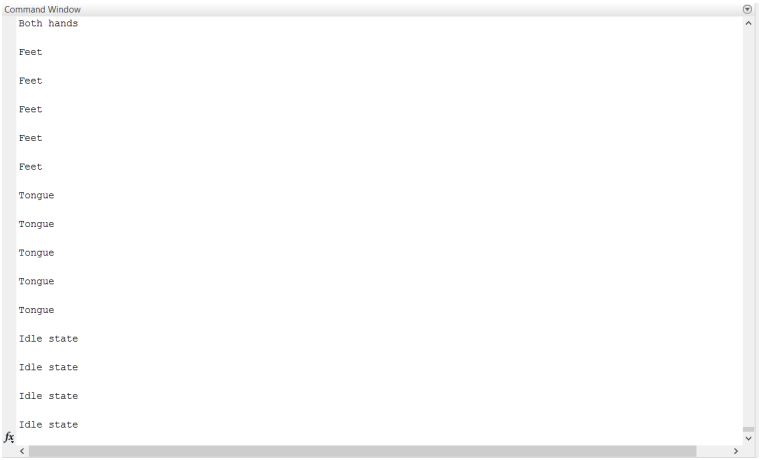


Figure 5.5: This figure displays an example of the feedback interface used doing real-time MI training. The text displays the MI performed at a constant pace.

Additionally, the real-time training system served as a way to investigate the performance and robustness of different classifiers built on the 6-class MI data sets obtained during data acquisition. The comparison of accuracy estimates determined from subject performance on this training system, was considered when selecting which classifier to apply in the resulting

BCI system.

5.3.2 Real-time simulation program

With the aim of providing the user with a more realistic training experience and ensuring that a given subject had trained sufficiently to be ready for controlling the drone in the real world, a virtual reality simulation was setup. This was accomplished by modifying a build-in VR demo from the MathWorks virtual reality toolbox, in which objects are represented in the virtual reality modelling language (VRML). The VR demo was transformed to enable it to take input from the developed MI-based BCI system. Furthermore, the VR simulation was modified in such a way that the 6-class BCI system would allow for as many degrees of freedom as possible.

The aircraft model used in the VR simulation was not a drone, but an airplane as the one seen in Fig. 5.6, as a drone was not available in the MathWorks toolbox. When launching the simulation program, the airplane commences on an airstrip, which is also visualized in the figure straight ahead from the plane. The airplane model advances or rotates a specific distance every time it receives a command, which signifies that it will not attain a constant velocity at any point and that it is only able to move or rotate in one direction at a time.

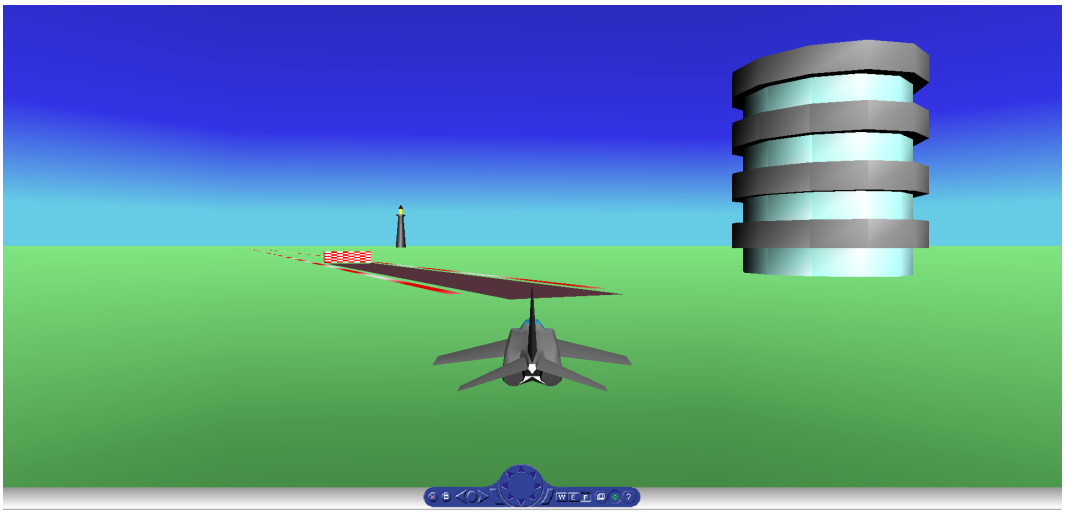


Figure 5.6: Virtual reality simulation program for aerial flight controlled by the developed BCI system. The shown VR simulation was generated by modifying a built-in VR demo from a MathWorks toolbox in Matlab.

The translation of which imagined movement corresponds to which airplane command can be seen in Table 5.3. Despite the fact that the aircraft is unable to pitch, roll or move backwards, it can be manoeuvred to any location in a 3D-coordinate system.

Tasks

To assess the subjects ability to control the airplane in real-time and to assist in determining the accuracy of the developed BCI system, two different tasks were designed.

The first task consisted in manoeuvring the plane through various, predetermined move-

ments with the aim of attempting to provoke all possible commands, i.e. all possible MI tasks. The specific planned route corresponded to the following:

1. Elevate the plane above the ground.
2. Fly forward to the roadblock on the airstrip.
3. Rotate 180° clockwise.
4. Fly back to the starting position.
5. Rotate 180° counter-clockwise.
6. Land the plane.

The next task was to fly the aircraft to a certain location by any self determined route after which the subject should return to the starting position. This task allowed the user to manoeuvre the plane in any desired manner without being forced to follow a predetermined series of tasks. Instructions for the specified task can be described as follows:

1. Fly to the top of the right-hand side building.
2. Fly back to the landing strip and land the aircraft.

The building in question can be seen on the right-hand side in Fig. 5.6.

Chapter 6

Signal Processing

This chapter elaborates on the various signal processing steps involved in the development of the resulting BCI system of this project. The decisions made with regard to the chosen methods and algorithms are discussed, alongside examples of the effect or results of the different processing steps applied. A section describing the different metrics used to assess the robustness of the developed BCI system in the result section concludes the chapter.

The processing of the EEG data is a crucial step in the design and development of a BCI system and, as described in Chapter 3, the data processing involves the following steps:

- **Preprocessing:** Raw EEG recordings are almost certainly contaminated with a combination of noise and artefacts. The purpose of preprocessing the EEG data is to remove this noise as well as minimizing the artefacts. In addition, a division of the signals into epochs also takes place in this processing step.
- **Feature extraction:** Relevant information characterizing each of the performed MI tasks can be extracted from the epochs. These characteristics are termed *features* and are the elements analysed by the classifier when distinguishing the MI tasks from one another. After extracting the features from the epochs, they are assembled in a vector termed a *feauture vector*, which is fed to the classifier. The dimensionality of the feature vector is therefore equal to the sum of features forwarded to the classifier.
- **Feature selection:** To assure the computational efficiency and a high classification accuracy, the combination of features describing the EEG signals must be properly selected. This implies that features which are either non-descriptive with regards to the chosen tasks or redundant with respect to other features, should be discarded, reducing the dimensionality of the feature vector. Furthermore, an investigation of the optimal combination of the remaining features was achieved through a program created in Matlab.
- **Classification:** In order to distinguish between the classes based on the selected features describing the EEG signals, a classifier scheme has to be selected and constructed. Assessment of the computational power needed and the classification accuracy achieved by a given classifier, was taken into account during the selection of an optimum classifier scheme for the BCI system in question.

6.1 Preprocessing

To prepare the EEG data for feature extraction and classification, a few preprocessing steps are necessary. These include dividing the data into small bits termed epochs, bandpass filtering the data to remove unnecessary frequencies and evaluating the necessity and means of performing artefact removal.

6.1.1 Division of the EEG data into epochs

Offline

The 6-class MI EEG data was recorded in sessions containing 12 trials of 10 seconds each combined to datasets of 60 trials, as indicated in Chapter 5. To prepare the data for feature extraction and classification, it was split into epochs of one second each. Furthermore, only the segment of each trial containing MI was divided into epochs and used to train the classifier. The latter corresponds to the time interval 4-10 sec of a trial.

To avoid EEG data containing the transition from the idling state to performance of the presented MI task, the first two seconds of the MI movement was removed. This was done based on the subjects' comments regarding the time it took for them to achieve the sensation of performing MI. This action resulted in the 6-10 sec section of each trial (see Fig. 5.3 for reference) being used to train a classifier.

In real-time

In the resulting BCI system of this project, the EEG data was processed in real-time, which modified the manner by which the data was divided into epochs. The epoch length of the online data was set to be identical to the epoch length of the offline data, namely, one second long. However, a half second overlap was added, resulting in the number of epochs for n seconds of recorded data being equal to $N_{epochs} = 2n - 1$.

The described overlap was made on the basis of increasing the BCI systems output rate. One second epochs without overlap would result in a command output once pr sec, which is slower than the desired output pace. With the overlap the BCI system forwards commands to the drone at a pace of one command pr. half second. Theoretically, this should on result in a risk of miss-classifying the transition from one MI action to another.

6.1.2 Filtering of the EEG data

As the recorded EEG signals are contaminated with noise and as the frequency range of interest is limited, the signals had to be filtered before further processing took place. To perform the necessary filtering a Butterworth bandpass filter was chosen to filter the EEG data.

Butterworth filters

The amplitude response of a nth order low-pass Butterworth filter is given by

$$|H(j\omega)_{low-pass}| = \frac{1}{\sqrt{1+(\frac{\omega}{\omega_c})^{2n}}}, \tag{6.1}$$

where ω corresponds to $2\pi f$ and ω_c corresponds to the cut-off frequency.

The Butterworth filters amplitude response decreases monotonically and the first $2n-1$ derivatives are zero at $\omega = 0$. These characteristics justifies why the Butterworth filter is sometimes referred to as a maximum flat filter. The attenuation of a low-pass filter at the cut-off frequency is always $-3dB$. Butterworth filters have a slow decreasing attenuation curve compared to other filter types, but is still often the preferred choice due to its monotonicity. For high values of the filter order n, the attenuation curve becomes steeper, leading to a smaller transition band and the amplitude response thereby approaches the amplitude response of an ideal low-pass filter. The latter is illustrated in Fig. 6.1.

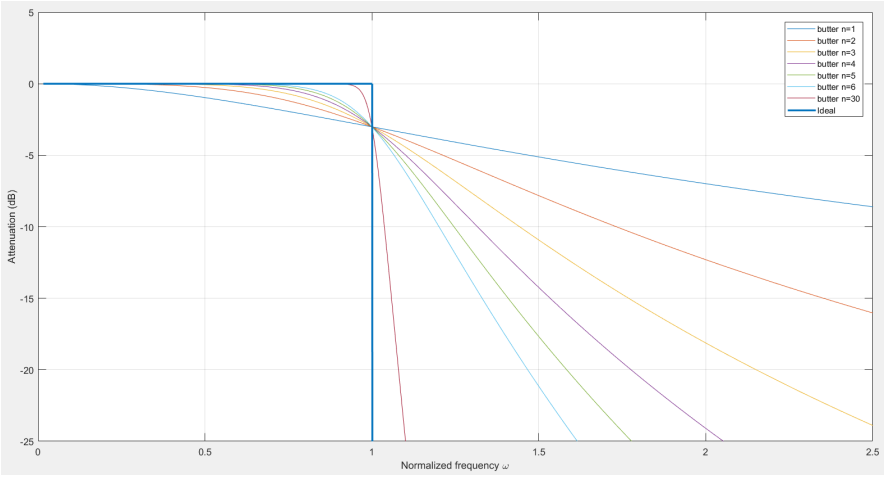


Figure 6.1: Comparison of the amplitude response of Butterworth filters of different orders.

As this project handles EEG data, it is necessary to filter out both high and low frequencies. A bandpass filter is therefore the obvious choice and such a filter can be realized by combining a low-pass and high-pass filter. A high-pass filter can be constructed from a low-pass filter by: $H_{high-pass} = 1 - H_{low-pass}$. However, as Matlab contains functions for constructing Butterworth bandpass filters, these are used for this project rather than constructing one from scratch. This is done on the basis that Matlabs own functions tend to be faster with regard to computational time required for processing.

A Butterworth bandpass filter of 4^{th} order, as used by Wendi Song et al. [46], with a bandpass ranging from 4 Hz to 25 Hz was created in Matlab and applied to filter the EEG data. This range was selected based on the fact that it contains the most important features for the MI movements used in this project. With the goal of avoiding phase distortion, zero-phase filtering was performed using the Matlab function `filtfilt` which filters the data forwards and backwards, resulting in a filtering of 8^{th} order. An example of the effect the applied filter had on a single trial can be seen in Fig. 6.2.

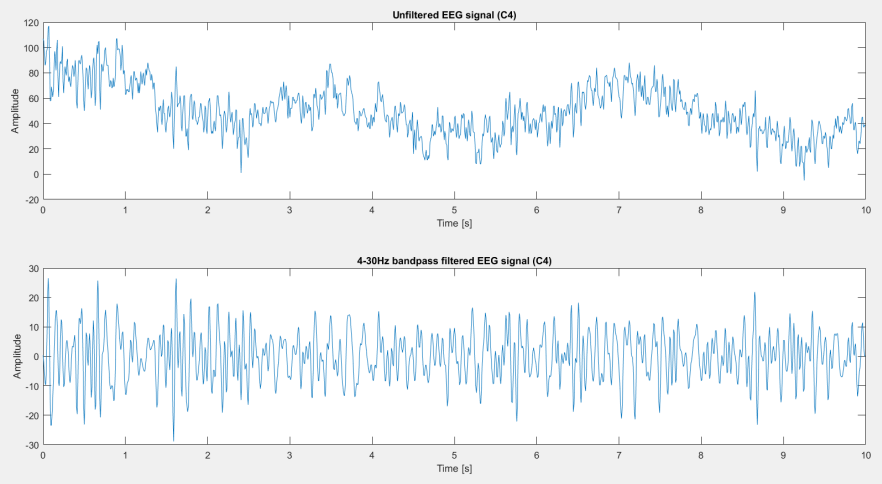


Figure 6.2: Unfiltered vs bandpass filtered EEG signal. The EEG data is from a single trial with a duration of 10 sec made with subject 2 and is taken from the C4 electrode.

6.1.3 Artefact removal

Artefacts, e.g. blinking and involuntary muscle movement, are generally identified in one of two ways:

1. Observing the subjects doing trials, while noting any blinks, eye or muscle movements and the time at which they occurred.
2. Manually looking through every trial and identifying every blink and movement from the EEG data itself or from scalp maps produced from the EEG data.

As the required personal for observing the subject during data acquisition was not available and as the time limit did not permit looking through all of the EEG data sets, no artefact removal was done prior to feeding the data to the classifier for training. The subjects were instead instructed to limit the generation of artefacts as much as possible and contaminated sessions were discarded based on instructions from the subjects, as mentioned in Chapter 5.

No artefact removal could be done on the EEG data when processing it in real-time, as no algorithms capable of decontaminating the data real-time has been developed to date [50]. In addition, it should be noted that artefacts such as blinking and eye movements are of such low frequency, that the bandpass filtering diminishes them sufficiently.

6.2 Feature extraction

An essential part of any BCI system, be it synchronous or asynchronous, is its ability to extract and quantify specific features of the recorded brain signals. As briefly described in the introduction of this chapter, the process of feature extraction corresponds to transforming the preprocessed signals, which were in epoch format at this point, into feature vectors. These vectors must be specific enough to enable the classifier to classify the given epoch as the right task and short enough to avoid overfitting and to reduce the amount of computational power in the

online system.

An abundance of different features have been used in the development of MI-based BCI systems. To provide the ability of estimating which features and in what combination these features would characterize the chosen MI tasks optimally, multiple feature extraction algorithms were developed to extract a variety of different features from the EEG signals. This ensemble of features was then passed through a feature selection algorithm, as will be described in the next section of this chapter. A summary of the 16 extracted features described in this section is provided in Table 6.1.

Table 6.1: Overview of the 16 extracted features described in this section.

Nr.	Feature	Nr.	Feature
1.	Activity (Hjorth parameter)	7.	Total average BP
2	Mobility (Hjorth parameter)	8	BP in the θ-band
3	Complexity (Hjorth parameter)	9	BP in the α-band
4	Log energy entropy	10	BP in the β-band
5	Shannon entropy	11	Spectral entropy
6	Zero-crossings	12	AAR parameter (5)

6.2.1 Time-domain features

Hjorth parameters

Introduced by Hjorth in 1970, the Hjorth parameters provide a swift way of computing three relevant characteristics of a time-varying signal, namely the activity, mobility and complexity. These parameters are very useful in BCI systems requiring fast computation of features because they are all based on variance [41].

The first of the three parameters, namely the activity, corresponds to the mean power of the signal, which is equivalent to the variance. It is therefore mathematically defined as:

$$\textbf{Activity} = \sigma_j^2 = \frac{1}{N}\sum_{n=1}^{N}(x_i(n) - \mu_j)^2 \tag{6.2}$$

where N is the number of data points in the epoch, $x_i(n)$ is the n'th sample of the i'th channel of the epoch and μ_j is the mean value of the j'th epoch.

The second of the Hjorth parameters, namely the mobility, is a measure of the root-mean-square frequency in a given epoch. It is mathematically defind as the square root of the variance of the derivative of the signal divided by the variance of the signal. This corresponds to the standard deviation of the derivative of the signal divided by the standard deviation of the signal.

$$\textbf{Mobility} = \sqrt{\frac{\text{var}(\frac{dx_i(n)}{dn})}{\text{var}(x_i(n))}} = \frac{\text{std}(\frac{dx_i(n)}{dn})}{\text{std}(x_i(n))} \tag{6.3}$$

The third of the parameters, namely the complexity, corresponds to a measure of the root-mean-square frequency spread. It attempts to reflect the deviation from the sine wave; a high

complexity means a high derivation from the sine wave. Therefore the degree of periodicity of the signal is captured by this feature. Mathematically, the parameter is defined as the mobility of the derivative of the signal divided by the mobility of the signal.

$$\textbf{Complexity} = \frac{\text{Mobility}(\frac{dx_i(n)}{dn})}{\text{Mobility}(x_i(n))} \tag{6.4}$$

Log energy entropy

According to a study published by Serap Aydin et al in 2009 [4], the log energy entropy values are considered as signal features that characterize the degree complexity in the EEG signal. It can be determined mathematically as follows:

$$E_{logenergy} = -\sum_n log_2(P(x_i(n)))^2 \tag{6.5}$$

where $P(x_i(n))$ is the probability of $x_i(n)$ showing up in the signal.

Shannon entropy

Introduced by Claude E. Shannon in 1948, the shannon entropy provides another measure for the complexity or uncertainty of the signal. The measure provides an indication of the predictability of the signal, as a high entropy means a high level of complexity in the signal and it is thereby harder to predict. It is defined as follows:

$$E_{shannon} = -\sum_n P(x_i(n))log_2(P(x_i(n))) \tag{6.6}$$

where $P(x_i(n))$ still corresponds to the probability of $x_i(n)$ showing up in the signal,

Zero crossings

This feature represents the total count of zero-crossings in an epoch. Zero-crossings corresponds to the amount of times the signal amplitude changes sign, which means a zero-crossing happens every time the signal crosses the x-axis. This parameter therefore reflects the frequency of the signal to some degree, as a high amount of zero-crossings indicates a higher frequency.

6.2.2 Frequency-domain features

As the features presented below are extracted from the frequency domain, one must first transform the signal from the time-domain to the frequency-domain. This can be achieved by using the Fourier Transform (FT), or in this case the Discrete Fourier Transform (DTF) as the signal is discrete. The latter is defined as:

$$\text{DFT}(x_i(n)) = X_i(F) = \sum_{n=0}^{N-1} x_i(n)e^{-j2\pi Fn}. \tag{6.7}$$

Average band powers

Finding the average band power can be done by estimating the distribution of power in the frequency domain and the total band power of an epoch can be used as feature.

According to Parseval's theorem, the power of the EEG signal in a single channel i, can be computed from the DFT as:

$$BP_i = \frac{1}{N}\sum_{k=0}^{N-1}|X_i(k)|^2, \tag{6.8}$$

where N is the total number of data points in the DFT and k corresponds to the counter running through these data points. The total average band power was used as a feature.

The BP does not necessarily need to be computed over the full range of frequencies present in the signal, but can be determined for a specific range of frequencies, which corresponds to restricting N to the desired frequency interval. This is done by first applying a bandpass filter to isolate the desired frequencies and then squaring the resulting signal, as to only have positive values. Hereafter the average band power can be obtained through integration.

Considering the band power of each of the five (six) frequency bands present in EEG signals (see Table 2.1 for the specific band intervals) can be used to indicate the brain activity occurring in each of the given brain wave patterns at any given time. As the μ-rhythm has proven dominant during MI in the contralateral side of the sensorimotor areas of the brain, the average band power over the frequency range 8-12 Hz can be used as a good indicator of what side of the body the subject imagines movement of a hand. It is therefore expected to be a prominent feature. However, it was decided to take the band power of the whole α-band in as feature instead of solely taking the band power of the μ-band, as this might be a better indicator in a 6-class system.

Additionally, the band power of the other types of rhythms might aid distinguishing between the 6 different classes. As the EEG signals were bandpass filtered from 4 to 25 Hz, the band power of δ- and γ-rhythms are irrelevant as features. However, the average band powers of θ- and β-rhythms were also included as features.

Spectral entropy

The spectral entropy is yet another feature which describes the complexity of the signal. The measure can be used to estimate changes in amplitude in the power spectrum. It can be determined from the Power Spectral Density (PSD) of the signal, which in turn can be estimated by squaring the spectrum and normalizing the result by the number of bins, k:

$$\mathrm{PSD}(F_k) = \frac{1}{k}|X(F_k)|^2. \tag{6.9}$$

The calculated PSD then has to be normalized to have values in the range [0;1] so that it can be viewed as a Probability Density Function (PDF) with an integral equal to one:

$$p_k = \frac{\mathrm{PSD}(F_k)}{\sum_k \mathrm{PSD}(F_k)}. \tag{6.10}$$

The power spectral entropy can now be estimated from the PSD through a modified version of the Shannon entropy Eq. (6.6), which yields:

$$\text{PSE} = -\sum_{k=1}^{K} p_k log_2 p_k \tag{6.11}$$

The Power Spectral Entropy indicates the frequency distribution across the spectrum: a high PSE corresponds to en epoch with a flat spectrum, which implies that the frequencies are fairly distributed across the whole frequency spectrum of the epoch. If the PSE value is low on the other hand, it implies that the power is concentrated around one or a few frequency bins.

6.2.3 Time-frequency features

AAR parameters

Autoregressive (AR) models predict current signal measurements based on past measurements by making use of the fact that natural signals tend to be correlated over time or sometimes space. A traditional AR model uses a set of coefficients a_i, which are equivalent to the AR parameters, to predict these measurements.

$$x_i(n) = \sum_{m=1}^{p} a_m x_i(n-m) + \varepsilon(n) \tag{6.12}$$

In the above, $a_m = a_1, a_2, ..., a_p$ are the AR parameters, $x_i(n-m) = x_i(n-1), x_i(n.2), ...x_i(n-p)$ is the delayed data points in channel i representing past measurements and $\varepsilon(n)$ is assumed to be a zero mean white noise process. The latter is added to account for the differences between the signal and its linear weighted sum approximation.

In the traditional AR model, a single set of coefficients are used, as the model assumes the statistical properties of the signal to be stationary. Brain-signals, however, tend to be non-stationary and the consequence hereof, is the requirement of a set of time-varying coefficients $a_{i,t}$. This consequence leads to an Adaptive Autoregressive (AAR) model, which can be described as:

$$x_i(n) = \sum_{m=1}^{p} a_{m,t} x_{t-i}(n-m) + \varepsilon_t(n) \tag{6.13}$$

The time-varying coefficients $a_{i,t}$ capture the signals local statistical structures as it evolves in time and these coefficients or parameters are used as features.

The parameter p represents the order of the AAR model and has to be selected carefully; choosing too low an order will cause the spectrum to smear, corresponding to a low spectral resolution, and choosing to high an order might result in modification of the original spectrum in the form of new peaks. The choice of order can either be done through an optimization process, e.g. cross validation or fixed a priori to a small arbitrary number according to the introductory text book on brain-computer interfacing by Rajesh Rao [41]. However multiple studies have

shown a fifth order model is adequate and even though a high order model might provide better result, a fifth order model was chosen to fit the scope of the project. This resulted in 5 AAR parameters, which was used as features and these were expected to provide good results, as multiple of the reviewed studies using these, which generally resulted in high classification accuracies (see Table 4.1).

6.3 Feature selection

16 different feature extraction methods have been portrayed in the previous section, with the aim of potentially describing the traits for each of the 6 classes. However, the amount of data needed to describe the classes increases exponentially with the number of features, which corresponds to the phenomenon often called "*the curse of dimensionality*". This gives rise to the necessity of eliminating some of the features to decrease the dimensionality of the feature vector.

Selecting the best features for a system with multiple classes can be achieved in a multitude of ways. The feature selection should be done with great care, as the extraction of well-describing features is crucial to produce high accuracy rates. One should for instance be aware of the fact that removing or adding a feature may not increase or decrease the overall accuracy consistently. So rather than removing a feature, observing a better classification accuracy and conclude that the feature is not worthwhile, specific combination of features should be investigated. Therefore, the individual accuracy corresponding to every possible combination of features should be computed to determine the best feature combination. It should come as no surprise, that this method is highly demanding with regards to computational power and thus results in a very time consuming process.

Other less computational demanding methods to select features exist. One method is to preselect features based on literature reviews and studies with similar classes as the desired BCI system, where it has been proven that the given features are able to characterise the aforementioned classes. Applying this method for selecting features will however increase the risk of choosing suboptimal features and as a result decrease the overall performance of the resulting BCI system. Furthermore, an overwhelming amount of features has been used throughout the literature for each class and paradigm and it would therefore be necessary to find the optimal feature for ones system by trial and error.

A second option is to perform a forward feature selection. In this method, the most favourable feature with regard to characterizing the classes of the system is determined from the list of features. Another feature is then added to the feature vector and the system is evaluated once more. If the second feature causes the accuracy to decrease the feature is discarded, on the other hand if the accuracy increases the feature is preserved and a third one is added to the feature vector and the system is evaluated once again. The algorithm applied in this context continues to add, discard and evaluate until there are no more features available on the list.

A third method of selecting features is to perform backward selection, where the accuracy of the BCI system is first assessed when all features are in use. Hereafter one of the feature is removed from the features vector and the accuracy of the system is determined again. If the accuracy improves the feature is then discarded and if it decreases, the feature is put back into the feature vector. The algorithm performing the backward selection continues in this manner until all features have been evaluated.

Both the forward and backward feature selection suffers from the deficiency that they do not evaluate all possible combination of features. The obvious advantage of these methods is that they require substantially less computational power than calculating the accuracy of all possible combinations of features.

Based on the aforementioned options, it was established that the optimal method for feature selection should be used and all combinations were therefore examined to ensure the quality of the developed BCI. To accomplish this goal, a Matlab script was written which could perform the necessary calculations. The number of combinations for a given list of features can be determined statistically from the dimensionality of the feature vector, as follows:

$$Combinations = \sum_{n=1}^{N} \frac{N!}{n!(N-n)!}, \tag{6.14}$$

where N is the number of features. The implications hereof is that for $N+1$ features, the number of combinations It will be increased to $2 \cdot It + 1$.

To evaluate the accuracy of each combination, a linear and a quadratic SVM classifier model with 5-fold cross-validation were applied, which will be described in further detail in the next section in this chapter.

As classifiers may vary in accuracy for each time it is used, the Matlab program was set up to classify the data five times, evaluate the five accuracies and then determine the mean of the latter to get a more precise estimate of the accuracy with standard deviation indicated. An example of the output generated by the Matlab program can be seen in Fig. 6.3.

FeatureColumn1	FeatureColumn2	FeatureColumn3	FeatureColumn4	Accuracy	std
1	2	3	4	' 58.75'	' 1.7541'
1	2	3	0	'54.6667'	' 1.9558'
1	2	4	0	'55.4444'	' 1.5416'
1	3	4	0	'56.5556'	' 1.1923'
2	3	4	0	'56.3889'	' 2.0624'
1	4	0	0	'53.1944'	' 1.8029'
1	2	0	0	'51.7778'	' 2.004'
1	3	0	0	'53.1389'	' 1.5732'
3	4	0	0	'52.1944'	'0.99381'
2	3	0	0	'53.6667'	' 1.384'
2	4	0	0	'52.3889'	' 1.854'
3	0	0	0	'52.0833'	' 1.7152'
1	0	0	0	'51.8056'	' 1.2767'
4	0	0	0	'48.4722'	'0.29463'
2	0	0	0	'50.6111'	' 1.1769'

Figure 6.3: Output table generated by the developed feature selection function, evaluating combinations of 4 features with three 6-class data sets from subject 2. The standard deviation in the table is denoted as std rather than SD. The applied classification model was a quadratic SVM and the features corresponded to **1:** Band power in the α-band, **2:** Hjort parameter 'Complexity', **3:** Band power in the β-band, **4:** AAR parameter nr. one.

In the provided example, only the combinations of four different features were evaluated by the program, as the table produced by the Matlab script with all 16 features was to long to present in the thesis. The total number of combinations in the latter, i.e. number of combinations when evaluating 16 features, is:

$$Combinations = \sum_{n=1}^{16} \frac{16!}{n!(16-n)!} = 65,535 \tag{6.15}$$

As seen in Fig. 6.3 the estimated accuracies are often quite similar. It is therefore important to keep in mind that the dimensionality of the feature vector has implications for the systems efficiency and assess the optimal feature combination by evaluating both accuracy and number of features. This implies, that if the feature vector which attained the maximum accuracy has a higher dimensionality than another with very similar accuracy and a small standard deviation, the feature vector of lower order dimensionality is preferred.

6.4 Classification

As mentioned in Chapter 4, machine learning has played an important role in the development of BCI systems by contributing with techniques that can learn to link cerebral activity recorded with EEG to suitable control commands. The algorithms for machine learning can be generally separated into two categories, namely, supervised and unsupervised learning. In supervised learning, sometimes termed function approximation, the given training data consists of a set of inputs labelled with their corresponding output and the goal is to learn the underlying function from this training data, so that new unlabelled input is classified as the correct output. Depending on whether the outputs are continuous or discrete classes, the problem is called either *regression* or *classification*, respectively. In the other category of machine learning algorithms, namely, unsupervised learning, the training data consists of unlabelled inputs and the goal is to learn a statistical model that may prove useful for ensuing analysis. Thus, unsupervised learning accentuates findings of hidden statistical structure in unlabelled data [41].

As the goal of this project was develop a BCI system that recognizes specific MI tasks and categorizes these correctly with regard to the given control commands, supervised learning was the obvious choice. Therefore the training data was recorded synchronously, so the set of inputs were labelled prior to feeding them to the selected machine learning algorithm. As the EEG signals are processed in one second epochs, the control commands are in the form of discrete classes, which leads to the conclusion that the necessary machine learning algorithm is a classifier and thus the problem is termed *classification*.

To reiterate, the objective of a classification algorithm is to assign one of N labels to a new input signal, given labelled training data sets consisting of known inputs and their corresponding output labels. According to a review of classification algorithms for EEG-based BCIs presented by F. Lotte et al. in 2007 [30], the following definitions are commonly used when describing the different kinds of classifier available:

- **Generative and discriminative:** Generative classifiers, also known as informative classifiers, learn the class models. The classification algorithm computes the probability of a feature vector belonging to each class and chooses the most probable. An example hereof is the Bayesian quadratic classifier. The discriminative ones, e.g. SVM, only learn how to discriminate the classes in order to classify a feature vector directly.
- **Static and dynamic:** Static classifiers, e.g. Multilayer Perceptrons, classify a single feature vector and thus cannot take temporal information into account during classification. Dynamic classifiers on the other hand, are able to classify a sequence of feature vectors and can thereby catch the temporal dynamics. An example of the latter is the Hidden Markov Model.
- **Stable and unstable:** Stable classifiers, such as LDA, have a low complexity and are said to be stable, due to their reaction to variations in the training data, as this does not con-

siderably affect their performance. On the contrary, unstable classifiers, e.g. Multilayer Perceptrons, have high complexity and small variations in the training data may cause important changes in performances.

- **Regularized:** The careful control of the complexity of a classifier with the aim of preventing overfitting is termed regularization. A regularized classifier is thus more robust with respect to outliers and has a good generalization performance.

Apart from the definitions given in the list above, classifiers are categorized as linear or non-linear and multiple studies have been conducted to determine which of these methods are superior with regard to BCI systems. Generally, linear classifiers are more robust due to their limited flexibility, i.e. fewer free parameters to tune, and are thus less prone to overfitting [32] [35]. However, non-linear methods can provide better results in some applications, especially when the application involves complex and/or very large data sets.

6.4.1 Selecting an appropriate classifier

Within BCI systems based on the MI paradigm, a variety of classification algorithms have been applied, with LDA, SVM and NN being some of the most widely used [32]. See 4.3 for a short introduction to these three kinds of classifiers.

Based on the literature review conducted during this project, it was decided to use the discriminative SVM classifier to solve the classification problem, as it has performed well in multiple other MI-based BCI systems and has several advantages suiting the aims of this project.

First and foremost, two major advantages of the SVM are its flexibility and simplicity in usage, as mentioned in 4.3. Moreover, the SVM is a regularized classifier and as the features extracted from the EEG signals often contains outliers, the regularization of the classifier is desired to provide robustness with regard to these outliers by increasing the generalization capacity of the classifier. In general, regularized classifiers have also proven to provide better results than their non-regularized counterparts and this is particularly the case for SVM [30]. SVM also has a low variance, which may be key to a low classification error in BCI, as BCI features are very unstable over time [32]. Many of these advantages come at the expense of a longer computational time than most other classifiers, but as it has priorly been deemed fit for online systems, it is assumed that it will not add any significant time-delay to the developed BCI system.

Furthermore, SVM is a stable classifier and will thus not be affected by variations in the training data. SVM is also a static classifier, but as it was only necessary to process one feature vector at the time in the developed system and as the temporal information is irrelevant to the classification in this system, this was deemed unproblematic. Finally, the SVM classifier has proven very robust to feature vectors of high dimensionality, implying that the classification algorithm handles the curse-of-dimensionality well. The latter is relevant to the developed system as it was decided to use multiple features and was not known whether the amount of acquired training data was sufficient in relation to the chosen number of features.

6.4.2 Support Vector Machines (SVMs)

Some classifiers, i.e. LDA and perceptrons, select a hyperplane $\mathbf{w}^T\mathbf{x} + w_0 = 0$, where $\mathbf{w}$ is the hyperplanes normal vector and w_0 is the threshold determined from the labelled data, to separate two classes. The chosen hyperplane is only one out of a potentially infinite number of hyperplanes dividing the two input classes. It can be demonstrated that among such hyperplanes, the best generalization is accomplished by selecting the hyperplane with the maximal margin between the two distinguishable classes [41].

The SVM is a classifier that determines exactly this, i.e. it selects the separating hyperplane for which the margin between the training data points of the two classes is maximized, as illustrated in Fig. 6.4.

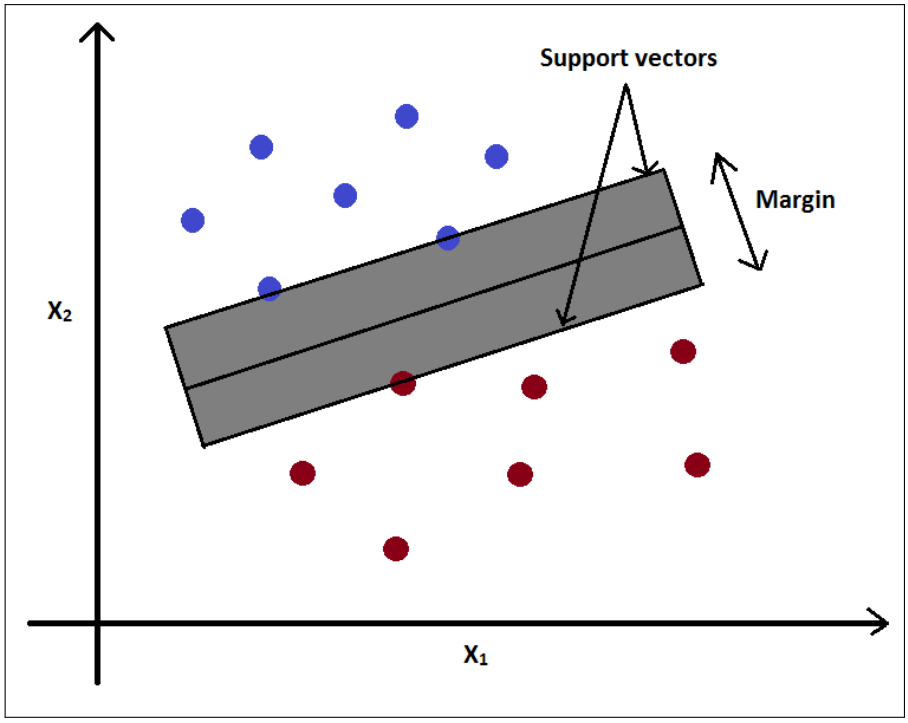


Figure 6.4: Illustration of a SVM finding the separation line with maximum margin between 2 classes, indicated as the red and blue dots. X_1 and X_2 denotes features. This figure is a modified version of Fig. 5.3 from [41].

As training data is often contaminated with noise and artefacts, outliers may appear in the feature space that makes it impossible to construct a hyperplane that separate the classes completely. In such cases, a soft margin SVM can be used which finds the maximal margin while keeping the number of misclassification to a minimum. The optimization problem for the linear soft margin SVM can be presented as:

$$\min_{\mathbf{w},\xi,w_0} \left[\frac{1}{2}||w||_2^2 + \frac{C}{K}||\xi||_1\right] \tag{6.16}$$

subject to:

$$y_1(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i$$

with $\xi_i \geq 0$ for $i = 1, ..., K$.

Where $||.||_2$ represent the Euclidian (L2) norm and $||.||_1$ the L1. ξ is the slack variable, that is a measurement of the degree of misclassification. $\mathbf{x}_i$ denotes the feature vectors. K is the number of input and y_i is the class membership. This equation is taken from *Brain-Computer Interfacing* by Rajesh Rao [41].

As features extracted from EEG data may be of high complexity, it is assumed that a linear SVM may not be sufficient and as such, a non-linear SVM may be utilized by using the *kernel trick* [41], which allow the SVM to achieve non-linear mapping as it employs the use of inner products between feature pair instead of computing each features coordinates. An example of how a non-linear SVM might work can be seen in Fig. 6.5.

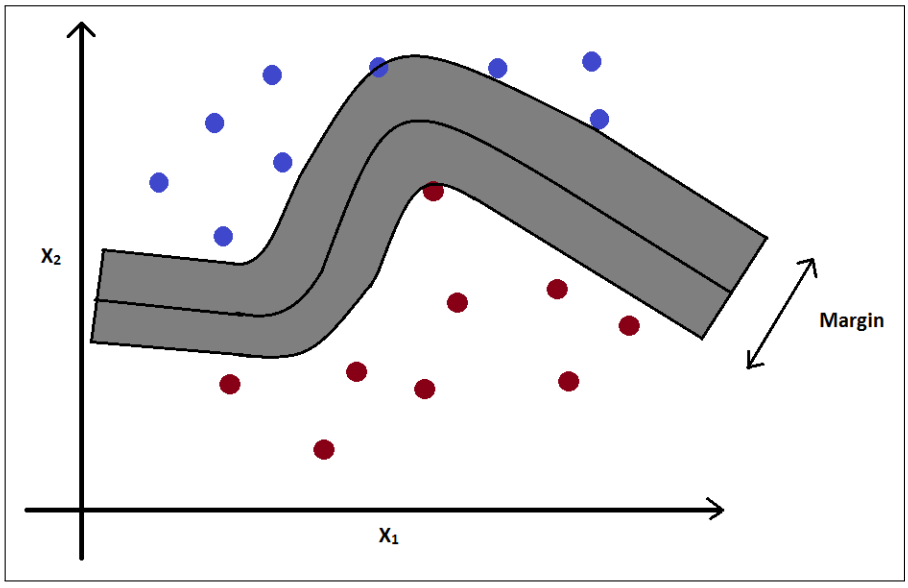


Figure 6.5: Illustration of a non-linear SVM finding the separation line with maximum margin between 2 classes, indicated as the red and blue dots. X_1 and X_2 denotes features.

Non-linear SVM models of kernel level 2 (quadratic) and kernel level 3 (cubic) are used in this project, as to examine how well they perform. Quadratic SVM have proven to be superior in some situations versus linear. Cubic usually provides higher accuracy versus both linear and quadratic, but have a high risk of overfitting the data.

6.4.3 Training the classifier

In multiclass machine learning, and SVM specific, there are two binary strategies for training a classifier: One-Against-One (OAO) or One-Against-All (OAA). In OAO the classification problem is reduces to a binary problem, where it will learn to differentiate one class versus another implying that a total of $\frac{N(N-1)}{2}$ binary SVMs will be trained, where N indicates the number of classes. In OAA the problem is also reduces to a binary problem, where the classifier will learn to differentiate one class versus all other classes giving a total of N SVMs. Even though OAO

requires more SVMs than OAA, it may create less support vectors than OAA as shown in Borra et al. [34]. As the the computation power required depends on the number of support vectors created, the OAO would require less training time than OAA. In this project, it was decided to use the OAO strategy as it is less complex than the OAA and because of the reduced training time, even though OAA is more commonly used [7].

As the data to be handled by the classifier was recorded synchronously, implying that the class of each epoch is known, a classifier can be trained using supervised learning. To achieve this, Matlab's Classification Learner was used, which is an integrated application for training a number of predetermined classifier types. Using this tool enables training of a wide array of different classifiers quickly and with relative ease and thereby enables testing different classifiers to determine the best one to use for a given type of data. An example of the Classification Learners interface and use can be seen in Fig. 6.6.

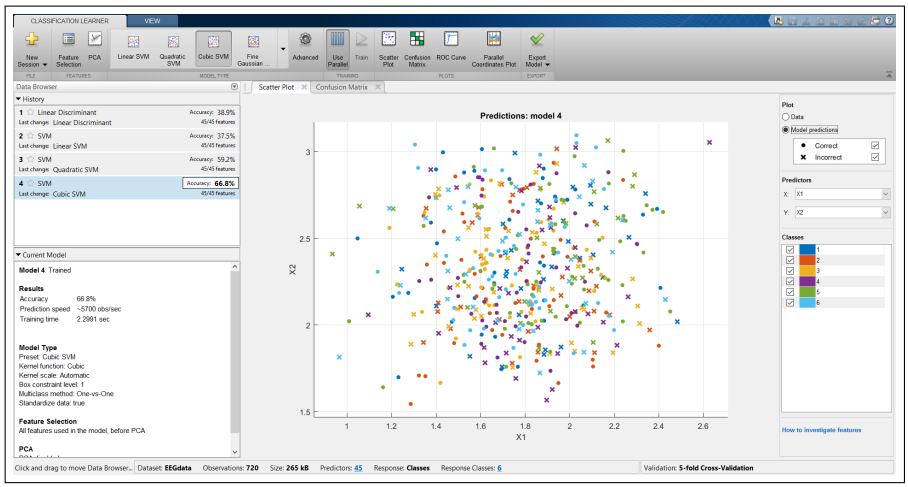


Figure 6.6: Matlab's Classification Learner in use with three 6-class data sets from subject 2. Accuracy of trained classifiers can be seen on the left side, while data point can be seen in the middle and colour-coding for the data point is on the right.

k-fold Cross-Validation

In this project, k-fold cross-validation was used to train classifiers and estimate the accuracies of these.

In k-fold cross-validation, the dataset is scrambled and partitioned in k bins. A classifier is then trained on $k-1$ bins thus leaving one bin out, which is then used for estimating the performance of the newly trained classifier. This process is then repeated k times until all combinations have been examined and the k accuracy results can then be averaged to reveal the estimated accuracy of the k-fold cross-validation. This method protects from overfitting the data and is a commonly used statistical means for estimating the classifiers performance on unlabelled data. A more precise variant of the k-fold cross-validation exists, the stratified k-fold cross-validation, which partitions the data in bins with even distribution of each label. As this is available through Matlab, this variant is used.

Choosing the right value of k can prove difficult, as higher values of k may produce estimates with lower variance but will require more data and computations. Lower values of k can increase

the variance due to instabilities of the training sets, especially in sets with many categories. It is commonly recommended to use either 5, 10 or 20 as values of k [24] [5]. It was therefore decided to test classifiers with 5 and 10 as values for k.

6.4.4 Evaluation of the BCI systems classification performance

As mentioned in the state-of-the-art, there are multiple evaluation techniques for quantifying the classification performance of asynchronous BCI systems and no official method has been decided upon. It was therefore necessary to determine by which metrics the developed system should be evaluated. Apart from the obvious parameter for performance evaluation, namely, the accuracy of the classifier, another important aspect with regard to the system developed in this project, is the classification time. The latter is important due to the fact that the resulting BCI system of this project is an online system and should compute commands from EEG signals fast enough, as to minimize the delay from thought to action in the drone. Therefore the classification time of the system will be taken into account, when presenting and discussing the results obtained from the BCI system. Furthermore, it was decided to use confusion matrices to illustrate the system performance, alongside the aforementioned accuracies estimated by the k-fold cross-validation, as these provide a deeper knowledge of the classifiers performance than simply indicating accuracy.

Confusion matrix

A confusion matrix provides information concerning the classification results of a BCI system in an illustrative and sensible manner. It is a $N \times N$ matrix, where N denotes the number of classes in the evaluated system, and the rows in the matrix represent the true class labels while the columns represent the classifiers output.

The case of the confusion matrix for a 2-class problem (*binary classification*) is illustrated in 6.7 and the four entries of the matrix corresponds to: the number of true positives (TP) or correct positive classifications, the number of false positives (FP) or incorrect positive classifications, the number of false negatives (FN) or missed positive classifications and the number of true negatives (TN) or correct rejections. The false positives or sometimes referred to as Type I errors, whilst the false negatives are sometimes referred to as Type II errors [41].

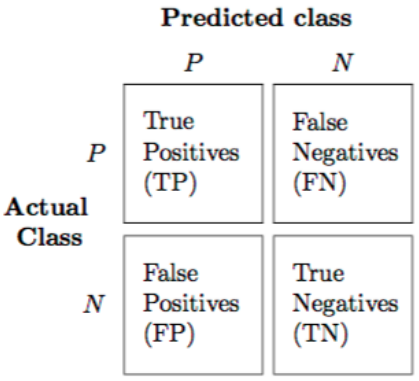


Figure 6.7: Confusion matrix for 2-class problem. P stands for *positives*, while N corresponds to *negatives*. Figure borrowed from [42]

From the matrix' diagonal, the total number of correctly classified samples can be determined as the sum of the diagonal elements. Furthermore, the non-diagonal elements of the confusion matrix illustrates the amount of samples misclassified by the BCI systems classifier.

The classification accuracy is defined as the ratio between correctly classified samples and the total number of samples provided to the classifier. It can thereby be derived from the confu-

sion matrix in the following manner:

$$\text{Accuracy} = \frac{TP+TN}{TP+FN+FP+TN}. \tag{6.17}$$

As the confusion matrices presented in the results of this thesis are computed by Matlab, they will be based on the k-fold cross-validation used to train the classifier and the accuracy determined by the above equation is therefore expected to be equal to the accuracy estimated by the k-fold cross-validation.

The error rate of the classifier can be determined by $\text{Error} = 1 - \text{Accuracy}$ and the chance level of an N-class system can be calculated as $\text{Chance level} = \frac{1}{N}$, provided that the number of examples for each class is the same.

Chapter 7

Results

This chapter presents the results obtained during this study. First, the results computed by the developed feature selection algorithm described in Section 6.3 are presented.

Secondly, the accuracies and confusion matrices computed for the 6-class system during offline classification for both both subjects are displayed. Furthermore, the system was reduced to a 2-class system, to verify the subjects' abilities with regard to performance of MI.

Finally, the results from the online classification and training are presented. Due to time limitations, the training phase was cut short and there were therefore no accuracies revealing the performance of the online system available to present. Therefore, a qualitative analysis of the accomplished part of the training phase is presented.

7.1 Feature selection

As described in Section 6.3, a feature selection algorithm was developed with the aim of determining the optimal combination of features to use in a 6-class BCI system. There was a total of 16 different extracted features to be examined by the algorithm, which corresponds to a total of 65,535 combinations as determined in Eq. (6.15).

Due to the computational power necessary for evaluating each sequence of features, computing the accuracy achieved for each combination of all 16 features was found to be to time-consuming with the available hardware. Therefore, it was decided to remove some of the elements from the feature vector prior to running the feature selection program. The collection of features to be removed were determined based on their relevance with regard to the performed MI tasks and their redundancy with regard to the other features. Three features were discarded, namely, zero-crossings, total average band power and the band power in the θ-band, corresponding to features six, seven and eight in Table 6.1.

Eliminating three features resulted in a feature vector with 13 elements. The number of combinations to be evaluated by the feature selection program, was thereby reduced from 65,535 to the following:

$$Combinations = \sum_{n=1}^{13} \frac{13!}{n!(13-n)!} = 8,191. \tag{7.1}$$

By running the developed algorithm with even fewer features and noting the computational time necessary for the program to complete its evaluation of all combinations, it was possible to estimate the necessary time for evaluating all combinations of the 13 features. It was assessed that the feature selection algorithm would need approximately two days to evaluate the accuracy of all 8,191 combinations of the 13 features with the available hardware. This time frame was deemed acceptable for the scope of this project. Nevertheless, the process was still notably time-consuming and therefore it was decided to run the feature selection program only once on data obtained from subject 2. The resulting features were used for training the classifiers for both subject 1 and subject 2, due to the limited amount of time available.

The size of the output table computed by the algorithm was to large to include in the report, as it included 8,191 rows containing the feature combinations and their corresponding accuracies. The combination of features yielding the best accuracy as computed by the algorithm, is presented in Table 7.1. The highest accuracy obtained was 74,6% and multiple other feature combinations resulted in similar accuracies, but were of higher dimensionality than the selected combination.

Nr.	Feature
3	Complexity (Hjorth parameter)
9	Band power in the α-band
10	Band power in the β-band
12	AAR parameter 1
12	AAR parameter 5

Table 7.1: Overview of the five optimum features for the developed BCI system, as determined by the feature selection algorithm. The features are presented alongside their corresponding feature number, as they were indicated in Table 6.1.

7.2 Offline classification

In this section, the results computed during the offline analysis of the 6-class BCI system are presented for both subjects. Furthermore, the 6-class system was reduced to a 2-class system with the classes left hand MI and right hand MI, and the results related to this reduced system are shown in the end of this section.

As explained in Section 6.4, it was decided to use SVM to solve the classification problem. However, it was not determined whether to apply the linear, cubic or quadratic SVM, as it was decided to make this decision based on the stability and performance of the three types of SVMs. Accordingly, accuracies obtained from all three kinds of SVM are presented for comparison. Furthermore, each of the classifiers were trained using both 5- and 10-fold cross-validation, as to investigate which of these yields the highest accuracies for the developed BCI system. The latter leads to an evaluation on a total of 6 different classifiers for each of the two subjects. To avoid redundancy, it was decided only to display the confusion matrix for the classifier which obtained the highest accuracy for each subject, as the prediction distributions were similar in all

computed confusion matrices.

7.2.1 Evaluation of the 6-class system

The results for classification of the 6-class MI EEG data by the developed BCI system is presented here for each of the two subjects. To reiterate, the classes are MI of the right hand (class 1), MI of the left hand (class 2), MI of both hands (class 3), MI of feet (class 4), MI of tongue (class 5) and finally the idling state (class 6), as previously indicated in Table 5.3. As stated in Section 6.4.4, the BCI system is evaluated by its produces accuracies and confusion matrices.

Subject 1

The accuracies obtained from the 6-class MI data acquired from subject 1, are shown in Table 7.2 for all three types of classifier trained with both 5- and 10-fold cross-validation.

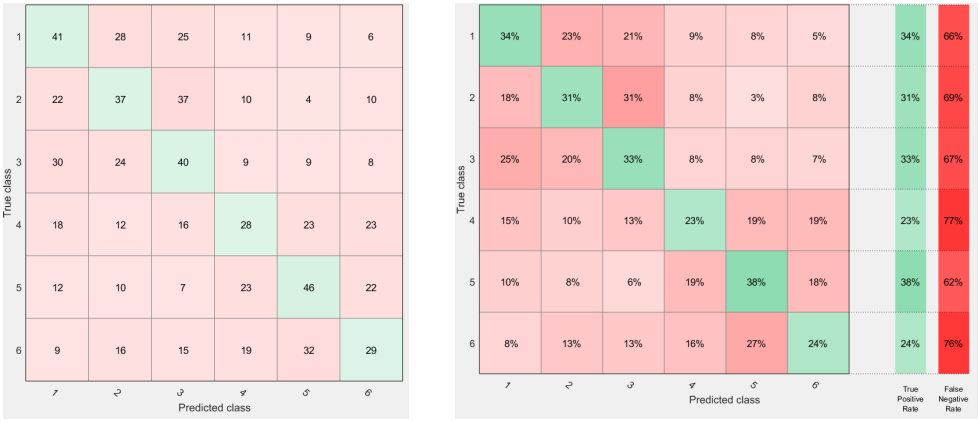
The accuracies obtained from subject 1 are generally low, but are still higher than the theoretical guessing level. There are no remarkable differences in the accuracies computed by the linear SVM, cubic SVM and quadratic SVM, respectively, although it seems as though the linear SVM performs slightly better then its two counterparts. Likewise, no noticeable differences are to be found between classifiers trained with 5- and 10-fold cross-validation.

Average accuracy	Classifier	k
30.1 %	Linear SVM	5
29.0 %	Quadratic SVM	5
27.2 %	Cubic SVM	5
30.6 %	Linear SVM	10
29.6 %	Quadratic SVM	10
27.2 %	Cubic SVM	10

Table 7.2: Table containing accuracies of trained classifiers on 6 classes. The data consists of three datasets split into 1 sec epochs from subject 1 and are the averages of 30 calculated accuracies with the corresponding classifier. The column **k** indicates the cross-validation used.

In Fig. 7.1, two versions of the confusion matrix computed from the classification of 6-class EEG data from subject 1 is displayed. From these confusion matrices, a clear tendency of misclassification of the different classes involving imagined hand movements can be observed, as the rates of false negatives and false positives are high around the aforementioned classes. Especially MI of the left hand is confused with MI of both hands, with class 2 being misclassified as class 3 as frequently as it is correctly classified.

Furthermore, the same tendencies can be observed with regard to class 4, class 5 and class 6, where especially class 5, i.e. MI of the tongue is confused with class 6, i.e. the idling state, with a false positive rate of 27 %.



(a) Number of observations **(b)** True positive/false negative rates

Figure 7.1: Confusion matrix of 6-class classification using linear SVM with data from subject 1 and a 10-fold cross-validation. The accuracy in the confusion matrix may vary from the accuracy in Table 7.2 as it is from a single training. Class 1: Right hand MI, Class 2: Left hand MI, Class 3: Both hands MI, Class 4: Feet MI, Class 5: Tongue MI, Class 6: Idle state.

Subject 2

The accuracies obtained from the 6-class MI data acquired from subject 2, are shown in Table 7.3 for all three types of classifier trained with both 5- and 10-fold cross-validation.

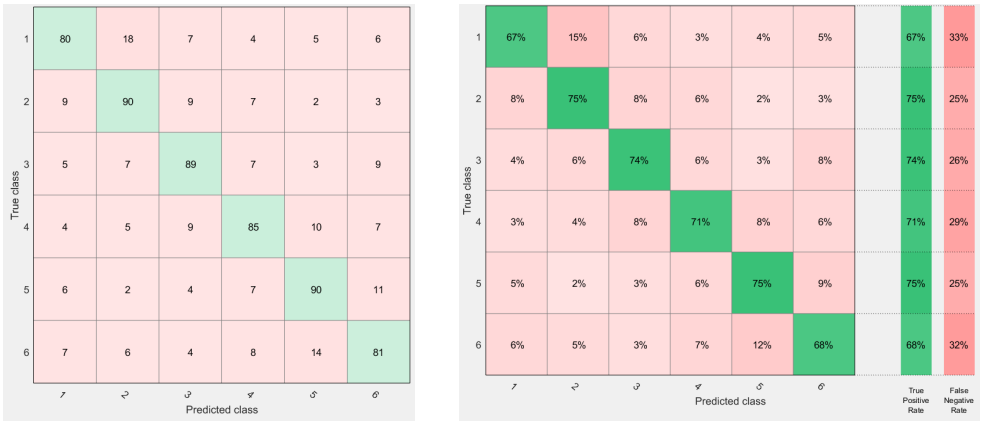
The accuracies obtained from subject 2 are substantially higher than the accuracies obtained from subject 1 with a maximum accuracy of 71,7 %. With regard to the classifiers, both the cubic SVM and quadratic SVM greatly exceeds the linear SVM in relation to the computed accuracies and the quadratic SVM seems like the best fit, when looking at the average accuracies. It can also be observed that all the classifiers trained with 10-fold cross-validation outperforms their counterparts which were trained with the 5-fold cross-validation.

Average accuracy	Classifier	k
39.7 %	Linear SVM	5
60.7 %	Quadratic SVM	5
66.7 %	Cubic SVM	5
40.8 %	Linear SVM	10
63.4 %	Quadratic SVM	10
71.7 %	Cubic SVM	10

Table 7.3: Table showing accuracy of trained classifiers on 6 classes. The data consists of three datasets split into 1 sec epochs from subject 2 and are the averages of 30 calculated accuracies with the corresponding classifier. The column k indicates the cross-validation used.

In Fig. 7.2, the same two versions of the confusion matrix presented with subject 1, are shown using the classification data from subject 2. As the accuracy is higher for subject 2 than subject 1, the misclassification tendencies are not as clearly distinguishable as in Fig. 7.1. However, some tendencies can be observed in the misclassification of class 1 as class 2, corresponding to the classifier misinterpreting MI of the right hand as MI of the left hand. Futhermore, a slightly elevated tendency of misclassifying class 6 as class 5 can be observed,

which corresponds to the classifier misinterpreting the idle state as MI of the tongue.



(a) Number of observations

(b) True positive/false negative rates

Figure 7.2: Confusion matrix of 6-class classification using cubic SVM with data from subject 2 and a 10-fold cross-validation. The accuracy in the confusion matrix may vary from the accuracy in Table 7.3 as it is from a single training. Class 1: Right hand MI, Class 2: Left hand MI, Class 3: Both hands MI, Class 4: Feet MI, Class 5: Tongue MI, Class 6: Idle state.

7.2.2 Reduction to a 2-Class system

In the following section, the results computed when reducing the 6-class system to a 2-class system are presented for each subject. The two classes of this reduced system were chosen to be class 1 and class 2, corresponding to MI of right hand and MI of left hand, respectively. The purpose of this reduction of the system was to investigate whether the subjects naïve to MI-based BCI were able to perform MI prior to any training. With this aim in mind, a second 2-class system was developed for testing the EEG data from subject 1, as he obtained substantially lower accuracies than subject 2. This system used class 3 and class 6, corresponding to MI of both hand and idling state, respectively. The classification was done with the three priorly described classifier versions, but these were only trained with 5-fold cross-validation as opposed to both 5- and 10-fold cross-validation as done previously.

Subject 1

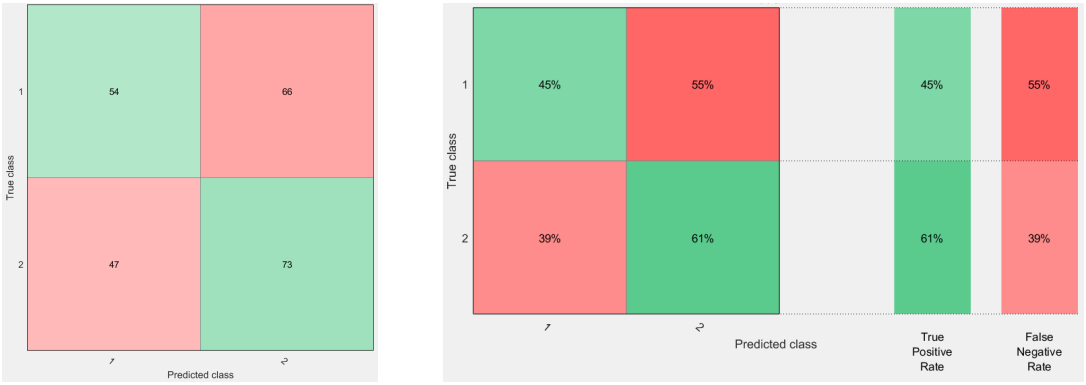
The average accuracies obtained from the 2-class system based on class 1 and class 2 with EEG data from subject 1 are presented in Table 7.4.

Average accuracy	Classifier	k
52.9 %	Linear SVM	5
50.4 %	Quadratic SVM	5
47.9 %	Cubic SVM	5

Table 7.4: Table showing accuracy of trained classifiers on 2 classes (Left- and right hand MI). The data consists of three datasets split into 1 sec epochs from subject 1 and are the averages of 30 calculated accuracies with the corresponding classifier. The column k indicates the cross-validation used.

The linear SVM outperforms the two other classifiers slightly, but as the accuracies approach the theoretical guessing level of 50 %, this does not indicate anything valuable.

In Fig. 7.3, the two versions of the confusion matrix computed from the classification of the 2-class system with subject 1 are presented. The most noticeable feature in these matrices is the misclassification of class 1 as class 2, with an astonishing false negative rate of 55 %, which implies that the MI of right hand movements was incorrectly classified more frequently than it was correctly classified.



(a) Number of observations **(b)** True positive/false negative rates

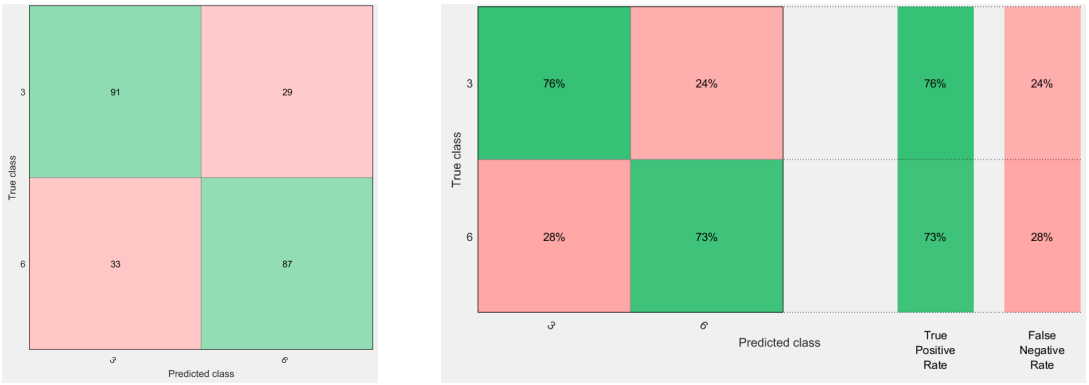
Figure 7.3: Confusion matrix of 2-class classification using linear SVM with data from subject 1 and a 5-fold cross-validation. The accuracy in the confusion matrix may vary from the accuracy in Table 7.4 as it is from a single training. Class 1 corresponds to right hand MI and class 2 corresponds to left hand MI.

In Table 7.5, the average accuracies obtained from the 2-class system based on class 3 and class 6 with EEG data from subject 1 are presented. Compared to the previously described 2-class system, the accuracies are much higher for this system and the linear SVM achieves the highest accuracy out of the three tested classifiers.

Average accuracy	Classifier	k
74.2 %	Linear SVM	5
67.5 %	Quadratic SVM	5
63.7 %	Cubic SVM	5

Table 7.5: Table showing accuracy of trained classifiers on 2 classes (Both hands MI and idle state). The data consists of three datasets split into 1 sec epochs from subject 1 and are the averages of 30 calculated accuracies with the corresponding classifier. The column k indicates the cross-validation used.

The two versions of the confusion matrix for the 2-class system distinguishing between class 3 and class 6 based on EEG data from subject 1 are shown in Fig. 7.4. The false prediction rate is quiet low, as the total accuracy is high compared to the other 2-class system with left/right hand MI. It should be noted that the true positives and false negatives of the confusion matrix are evenly distributed.



(a) Number of observations **(b)** True positive/false negative rates

Figure 7.4: Confusion matrix of 2-class classification using linear SVM with data from subject 1 and a 5-fold cross-validation. The accuracy in the confusion matrix may vary from the accuracy in Table 7.5 as it is from a single training. Class 1 corresponds to both hands MI and class 2 corresponds to Idle state.

In Fig. 7.5 the average power spectrum of right hand MI, left hand MI, both hands MI and idling state from a whole dataset is shown for subject 1. As it is hand MI, the most noticeable area should be the μ band (8-12Hz). For the right hand MI, it can be seen that for both the C3 and the C4 electrodes the power in the μ band is diminished compared to the idling state. The same is true for the both hands MI. For left hand MI, the C4 electrode is diminished whilst the C3 does not appear to be.

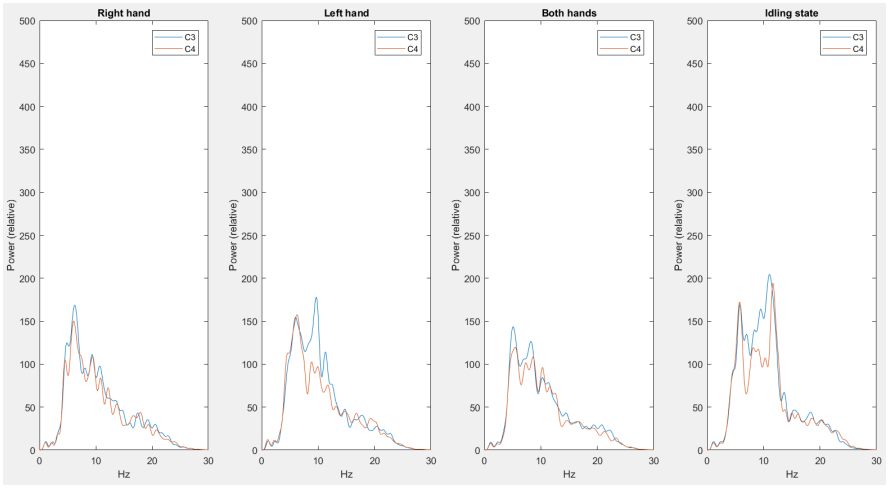


Figure 7.5: The power spectrum of left/right hand, both hands and idle state in the C3 and C4 electrode location. The data is the average power spectrum of a dataset from subject 1.

Subject 2

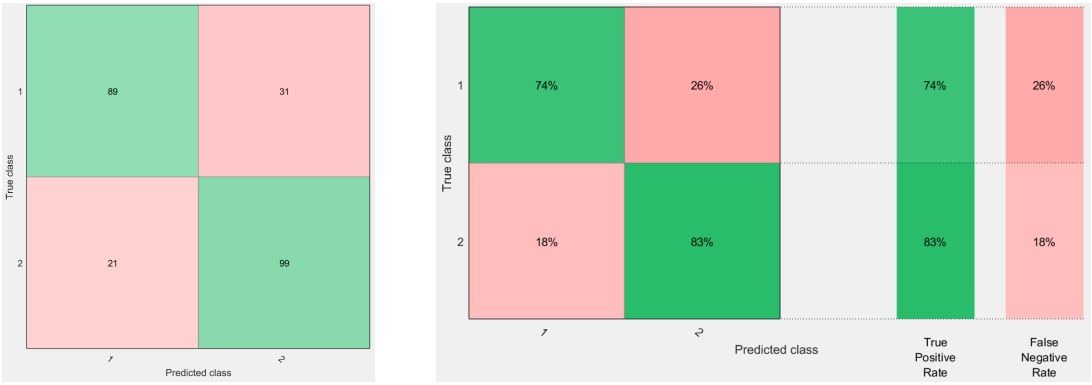
The average accuracies obtained from the 2-class system based on class 1 and class 2 with EEG data from subject 2 are presented in Table 7.4. Compared to the accuracies computed from subject 1's EEG data, a substantial elevation in the average accuracy can be observed with a maximum average accuracy of 80,4 %.

Average accuracy	Classifier	k
66.3 %	Linear SVM	5
78.3 %	Quadratic SVM	5
80.4 %	Cubic SVM	5

Table 7.6: Table showing accuracy of trained classifiers on 2 classes (Both hands MI and idle state). The data consists of three datasets split into 1 sec epochs from subject 1 and are the averages of 30 calculated accuracies with the corresponding classifier. The column k indicates the cross-validation used.

The quadratic and cubic classifiers outperform the linear SVM, which was also the case for the 6-class classification for subject 1, with the cubic SVM being slightly ahead of the quadratic SVM in obtained accuracy.

In Fig. 7.6, two versions of the confusion matrices for the 2-class system are displayed. The same tendency as for subject 1, i.e. right hand MI misclassified as left hand MI, can observed from the confusion matrix, although to a lesser degree.



(a) Number of observations **(b)** True positive/false negative rates

Figure 7.6: Confusion matrix of 2-class classification using quadratic SVM with data from subject 2 and a 5-fold cross-validation. The accuracy in the confusion matrix may vary from the accuracy in Table 7.6 as it is from a single training. Class 1 corresponds to right hand MI and class 2 corresponds to left hand MI.

In Fig. 7.7 the average power spectrum of right hand MI, left hand MI, both hands MI and idling state from a whole dataset is shown for subject 2. It can be seen that for right hand MI, the C3 electrode is much more diminished than the C4 electrode. The reverse is true for the left hand MI. For both hands MI, both C3 and C4 is diminished.

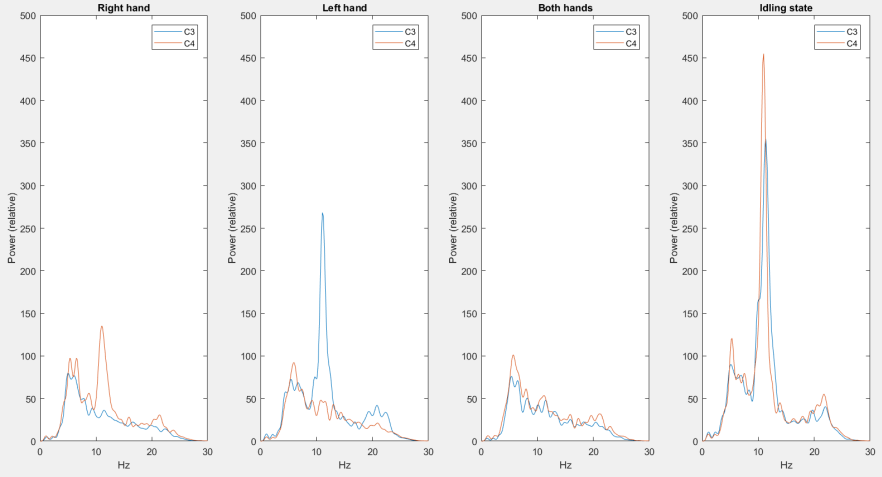


Figure 7.7: The power spectrum of left/right hand, both hands and idle state in the C3 and C4 electrode location. The data is the average power spectrum of a dataset from subject 2.

7.3 Online classification and training

The online training was performed using text based feedback with the setup described in Section 5.3.1. As mentioned in the introduction to this chapter, the training and online testing phase was cut short due to time limitations, which resulted in subject 1 being the only one of the subjects to experiment with the training system. As it was deemed necessary to train the subjects prior to them attempting the two tasks in the VR simulation described in Section 5.3.2, no attempts were made to complete these tasks. However, some observations were made during subject 1's experimentation with the training program.

All the different types of SVMs were tried out during the experimentation with the real-time training system, i.e. linear, quadratic and cubic SVM, as to evaluate the performance and stability of each of these classifiers.

Using the linear SVM for testing on subject 1, revealed that when the subject was performing any of the three hand MI tasks, the classifier would randomly categorize it as either class 1, class 2 or class 3, corresponding to MI of right hand, MI of left hand and MI of both hands, respectively, with a few offsets to the other classes. When the subject was performing either MI of the feet or MI of the tongue, the linear classifier categorized it randomly as either class 4, class 5 or class 6, corresponding to MI of the feet, MI of the tongue and idle state, respectively, with few offsets to other classes. When performing idle state, the subject was able to get the classifier to categorize it correctly with very few offsets.

Using the quadratic SVM for subject 1, the same tendencies as with the linear SVM were seen. It could not be concluded whether or not this provided a more stable results than the linear classifier.

Using the cubic SVM for subject 1, the same tendencies were also true as for the other two, though it were more unstable in classifying as it more frequently jumped between all classes.

Chapter 8

Discussion

In this chapter a discussion on the obtained results is provided. This includes a brief criticism on the feature selection process, a longer discussion on the estimated accuracies for the 6-class and 2-class system, respectively, and a short analysis of the online tests. A discussion on the future work and possible optimization proposals, concludes the chapter.

8.1 Feature selection

As the optimal set of features may differ from one subject to another, the feature selection algorithm should have been used to compute the optimum feature combination for each of the two subjects. However, as previously stated in Section 6.3, the computational power available was not sufficient and thus the time required for computing another optimal feature combination for subject 1 was out of the scope of this project.

Additionally, by only testing the algorithm with 13 out of 16 features, potential optimal combinations may have been lost in the process. With greater computational power available, like a cloud based hub system, it would be possible to check all combination for all 16 features and implement even more features if desired in a short amount of time, compared to using one personal computer for the task.

8.2 6-class

In Table 7.2 described in the previous chapter, the accuracies obtained from the 6-class system evaluated on the EEG data from subject 1 are displayed. The accuracies are all consistently low regardless of the classifier type, corresponding to approximately $\sim$29% which is almost double the theoretical guessing level of $\sim$16.67%. This does indicate some correlation but could also be a result of overfitting when training the classifier. The linear SVM appears to provide the highest accuracy and the degree of k-fold cross-validation does not affect the accuracies. When inspecting Fig. 7.1, it can be observed from the confusion matrix that the three hand MI tasks, i.e. left hand, right hand and both hands, often get misclassified with one another, while the same is true for MI of the feet, MI of the tongue and idle state. This would indicate that the

classifier is able to distinguish between MI tasks based on hand movements and other tasks, while it is not able to distinguish between the different MI tasks involving hand movements themselves.

The accuracies obtained from the 6-class system when evaluating on the EEG data obtained from subject 2, are much higher than that of subject 1 for all the concerned classifiers, as can be seen in Table 7.3. Unlike in the results for subject 1, the accuracy varies substantially with regards to the SVM degree with Linear being the most inaccurate at 39.7% and cubic being the most accurate at 66.7% when considering the 5-fold cross-validation results. As stated in Section 6.4, it has been suggested that a linear classifier might not be able to classify data of high complexity, which would explain the remarkable difference in accuracy of the linear- and non-linear classifiers for subject 2. There is a noticeable rise in accuracy when using 10-fold cross-validation compared to 5-fold as seen in Table 7.3 which is consistent with the reviewed literature [5] [24]. The highest accuracy achieved was with cubic SVM using 10-fold cross-validation at 71.7% which is greatly superior to the theoretical guessing level, suggesting a large variance between the different classes. From the confusion matrix in Fig. 7.2, it can be observed that the false prediction rates are close to evenly distributed between all classes, which indicates that non of the classes correlate too much with one another. However, as mentioned when describing the results, a minor tendency of classifying right hand MI as left hand MI can be observed, as well as an even smaller tendency of classifying idling state as MI of the tongue. The latter suggests that the subject may have found it difficult to perform MI of the tongue, resulting in similar brain patterns as observed during the idling state.

In 2014, Song et al. [46] measured accuracies of a 4-class BCI system on subjects over five consecutive days to see the improvement of training and the study showed that one subject had an increase in accuracy from 40% to 89.5% over these days. As the data recorded during this project, which was used for estimating the accuracies of both subjects, was recorded over a timespan of two days, the accuracy of 71.7% could be considered impressive when compared to the described study by Song et al., as it comes from estimating 6 classes without any prior training. With further subject training, it would be expected that this accuracy increases and it would not be a bad assumption that subject 1's accuracy could improve. It should be noted that the accuracies might not be completely comparable, as the ones from Song et al. are obtained with an online system as opposed to the offline estimate from this project.

The system developed by Doud et al. [10] shows a 6 class system achieving up to 87.2% from online testing. As above, this accuracy is obtained after subjects have trained performing the specific MI tasks multiple times, as Doud et al. emphasizes training as a main contributor for increasing accuracy in a subject.

8.3 2-class

The system was reduced to a 2-class system to see whether or not the left- and right hand MI could be predicted and thereby, if subject 1 was even able to perform this class of MI. From Table 7.4 it can be seen that the accuracy of left- and right hand MI is $\sim$50% and Fig. 7.3 shows that the distribution is not shifted towards one of the classes. This indicates that these classes can not be distinguished from each other as the predicted accuracy is near the same as the theoretical guessing level, which could mean that the subject is unable to perform MI or that the features does not represent distinctive characteristics of the performed MI. From Section 4.3

it is suggested that left/right hand movement produces ERD in the contralateral brain area in the μ band while performing this type of MI. Fig. 7.5 shows the power spectrum of left- and right hand MI compared to MI for both-hands and idle state at the C3 and C4 electrode positions, which corresponds to the areas where ERD would be visible for these MI tasks. For MI of both hands, this shows that the ERD appears for both electrodes, when compared to idle state. At the MI of left hand, ERD appears mainly in the C4 electrode and for the MI of the right hand, ERD appears for both electrodes as in the case with both-hands MI. A sound conclusion from this, would be that the subject is unable to perform left- and right hand MI separately. This might be due by the fact that the subject suffers from Tourette's Syndrom, which affects the motorsensori area of the brain, but as no previous studies have been made about this subject, it cannot be concluded to be the definite cause.

As stated previously in Section 8.2, the 6-class MI classifier for subject 1 appeared to classify into one of the three hand-classes when MI of the hand was performed, or into one of the other classes when one of these were performed. As such, a classifier was trained for 2-classes, i.e. MI of both hands and idle state. The result, which can be seen in Table 7.5, shows that the achieved accuracies were higher than that of the MI tasks of the left and right hand, with the highest accuracy being 74.2% when using linear SVM. This is significantly higher than that of the theoretical guessing level at 50%. Moreover, Fig. 7.4 shows that the distribution of true and false predictions are even over the two classes. As such, it can be concluded that the classifier is able to distinguish between these classes.

Though subject 2 was able to achieve high accuracies with the 6-classe system, a 2-class problem was assessed too, to enable comparison with the results of subject 1. A left- and right hand MI classification was attempted. The results can be seen in Table 7.6 with the highest accuracy being 80.4% using cubic SVM. Fig. 7.6 shows that the distribution of predictions is shifted slightly to favour left hand MI. Furthermore, Fig. 7.7 displays the power spectrum for the MI of the hands and the idle state in the C3 and C4 electrodes, in which it can be seen that ERD appears for both C3 and C4 for all hand MI but for right hand MI, the ERD is much greater in C3 than in C4 and vice-versa for left hand MI. For MI of both hands it can be observed that the ERD is of the same amplitude. This leads to the conclusion that subject 2 is indeed able to perform left/right hand MI. The fact that the distribution is shifted slightly towards left hand for both the 2-class and the 6-class systems is unexpected, as the subject is right handed and thus a shift towards the right hand would be expected.

The study by Tian et al. [49], where left/right hand MI was performed, showed an accuracy of up to 91.0% using wavelet transform for feature extraction and a RBF NN as classification method. Compared to this, the accuracy obtained by subject 2 of 80.4% is quite low. An explanation behind this could be the fact that the features selected were optimized for use in a 6-class system, meaning that some feature may be redundant and some of the removed may have improved a 2-class hand based system. Furthermore, the optimal electrode placements were not available on the modified cap using Emotiv's system, as the remaining electrodes surrounding the motorsensori cortex were missing. Additionally, if a 2-class system was to be the objective, new features would have to found.

8.4 Online tests

The online training with feedback showed that subject 1 was able to produce steady result when trying to provoke the idle state respond, although he was unable to do the same for any of the other classes. As Fig. 7.1 shows, there is a clear correlation between all the different types of hand-based MIs, which corresponds well with the feedback subject 1 got when performing any of these. The same can be seen with the other three classes, part from idle state, as feet and tongue MI most often would result in classification of feet, tongue or idle state. As the subject was only able to continuously provoke the idle state, it was determined that he should not continue to the live simulation at any point soon. Furthermore, it was also discussed whether or not he was MI illiterate, but no conclusion was found. It was decided that the subject should be tested to see if training could improve the accuracy to a sufficient level, as was achieved by Doud et al. [10] and Song et al. [46].

As limitations in time did not allow for online testing of subject 2, not much can be said about it. But as subject 2 did show notably higher accuracy in the offline evaluation than subject 1, it is expected that online testing with her will provide better result that might prove good enough for testing the live simulation and finally the drone. Online testing and training for subject 2 is expected to be done in the near future.

8.5 Future work

As this thesis only presents a proof-of-concept, the goal is the make the concept an applicable study. For this, at least ten subjects should be used for testing over an extended period of time to allow training, as multiple studies have shown that this can greatly improve accuracies for MI based BCI, best shown in the before mentioned Song et al. [46].

In the near future, it is planned to implement more features that to increase the chance of finding an optimal combination. One of such features would be a multiclass CSP (common spacial pattern) filter, as this is able to greatly increase the variance between classes. Another feature that might get implemented is the wavelet transform, as this have proven to be useful in classifying multiclass systems before [46].

Equipment wise, the system is expected to greatly improve if a cap with all relevant electrodes were used, as many features depends on the variance of electrodes close to each other. As the system was supposed to be wireless, a bluetooth transmitter with an effective range of more than 20 cm would be preferred, a transmitter using wi-fi would properly be optimal for this job. In the far future, when the system works on multiple subjects with minimal training, a Raspbarry pi might be implemented to bypass the need of a PC, which would make the system more mobil and flexible to use. As the main feedback when flying a drone would be a front-placed video camera, it would be optimal for the user to be able to turn it without the need to turn the drone itself. Eye-tracking or a gyroscope could be implemented, as these would only have little interference with the MI BCI system itself. All this could be combined with a microphone and a speaker to make long distance communication possible for user unable to move and thereby they might achieve a better quality of life.

Chapter 9

Conclusion

The aim of this project was to develop part of a hybrid BCI system permitting the control of a drone for paralysed patients, based on the hypothesis that this would equip the patient group with increased mobility and thereby elevate their quality of life. The system part in focus was the BCI capable of categorizing the MI tasks provided as input by the patient as control commands that could be fed to the drone with as high an accuracy as possible.

To enable control of a drone in 3 dimensions, a 6-class system was developed, as this is the minimum amount needed for accomplishing this task, with the movements corresponding to forward, turn left and right, move up and down, and stay still. As it was decided to apply MI as the paradigm of the BCI system, 5 different MI task were found in accordance to tasks that had previously been distinguished by other researches. The chosen MI tasks were as follows: Left and right hand, both hands at the same time, feet and tongue. Furthermore, the sixth class was selected to be the idle state.

In order to be able to classify the MI tasks, features needed to be extracted to characterize the tasks from one another. The features for this were found using an algorithm able to determine the combination of features that would provide the best result. From this, 5 features were selected to be used by all subjects. Three different classifiers were used to categorize the classes using the extracted features, namely, linear SVM, quadratic SVM and cubic SVM. EEG data sets of the aforementioned MI tasks were acquired from two subjects and the data was used for training the classifiers. The data sets consisted of 60 trials with 4 seconds of data each and three of these data sets were deemed sufficient for training the classifiers. The accuracy of the classifier was estimated using both 5-fold or 10-fold cross-validation.

The 6-class system was evaluated upon and one subject was able to achieve high accuracies up to 71.7% with even distribution of false predictions, whilst the other was unable to achieve higher accuracies than 30.6%, which would be considered to low for drone control. The system was reduced to a 2-class problem to evaluate if the subject was able to perform distinguishable hand MI. This however, showed that he was not able to get accuracies above the theoretical guessing level.

As the time did not allow for this study to finish testing, simulation and drone flight, the feasibility of this can not yet be concluded. This study does however, show that classifying 6 classes is possible.

Bibliography

[1] Mohammed J Alhaddad, Mahmoud I Kamel, and Dalal M Bakheet, Unknown upload data. URL `https://www.researchgate.net/figure/274831841_fig1_Figure-1-Basic-Block-Diagram-of-BCI-Systems`.

[2] K. K. Ang, Z. Y. Chin, C. Wang, C. Guan, and H. Zhang. Filter bank common spatial pattern algorithm on bci competition iv datasets 2a and 2b. 2012.

[3] Unknown author, Unknown upload date. URL `http://neuro.imm.dtu.dk/wiki/Beta_rebound`.

[4] Serap Aydin, Hamdi Melih Saraoglu, and Sadik Kara. Log energy entropy-based eeg classification with multilayer neural networks in seizure. *Annals of Biomedical Engineering*, 37(12), 2009.

[5] Simone Borra and Agnostino Di Ciaccio. Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods. *Department of Economics and Territory, University of Rome*, 2010.

[6] Alvaro Rodrigo Fuentes Cabrera. Ph.d. thesis: Feature extraction and classification for brain-computer interfaces. 2009.

[7] C.-Y. Chen, C.-W. Wu, C.-T. Lin, and S.-A. Chen. A novel classification method for motor imagery based on brain-computer interface. 2014.

[8] Z. Y. Chin, K. Ang, and C. Wang et al. Multi-class filter bank coomon spatial pattern for four-class motor imagery bci. 2009.

[9] Stefan Cososchi, Rodica Strungaru, Alexandru Ungureanu, and Mihaela Un gureanu. Eeg features extraction for motor imagery. *28th Annual International Conference of the Ieee Engineering in Medicine and Biology Society*, 2, 2006.

[10] Alexander J. Doud, John P. Lucas, and Bin He. Continuous 3d control of a virtual helicopter using a motor imagery based bci. *International IEEE/embs Conference on Neural Engineering*, 2011.

[11] O. Diana Eva and D. Tarniceriu. Substitution of spatial filtering from relaxation to motor imargey for eeg based brain computer interface. 2015.

[12] A.A. Frolov, E.V. Biryukova, P.D. Bobrov, O.A. Mokienko, A.K. Platonov, V.E. Pryanichnikov, and L.A. Chernikova. Princinples of neurorehabilitation based on the brain-computer interface and biologicalle adequate control of the exoskeleton. *Human Physiologi*, 39(2), 2013.

[13] C. Guger, G. Edlinger, W. Harkam, I. Niedermayer, and G. Pfurtscheller. How many people are able to operate an eeg-based brain-computer interface (bci)? *Ieee Transactions on Neural Systems and Rehabilitation Engineering*, 11(2), 2003.

[14] M. Hamedi, S.-H. Salleh, A. M. Noor, and I. Mohammad-Rezazadeh. Neural network-based three-class motor imagery classification using time-domain features for bci applications. 2014.

[15] Yasunari Hashimoto and Junichi Ushiba. Eeg-based classification of imaginary left and right foot movements using beta rebound. *Clinical Neurophysiology*, 2013.

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

[17] T. Hiroyasu, Y. Ohkubo, and U. Yamamoto. Electroencephalographic method using fast fourier transform overlap processing for recognition of right- and left-handed elbow flexion motor imagery. 2014.

[18] W.-Y. Hsu and Y.-N. Sun. Eeg-based motor imagery analysis using weighted wavelet transform features. 2009.

[19] Han-Jeong Hwang, Soyoun Kim, Soobeom Choi, and Chang-Hwan Im. Eeg-based brain-computer interfaces: A thorough literature review. *Intl. Journal of Human-Computer Interaction*, 2013.

[20] Alessandra Ibba. Introduction to brain-computer interface systems. 2016.

[21] Noppadon Jatupaiboon, Setha Pan-ngum, and Pasin Israsena. Real-time eeg-based happiness detection system. *Scientific World Journal*, 2013, 2013.

[22] J. Jin, X.-Y. Wang, and X. Zhang. Recognition of left and right motor imagery based on energy features. 2007.

[23] Christoph Kapeller, Christoph Hintermülle, and Christoph Guger. Augmented control of an avatar using an ssvep based bci. *AH'12 Proceedings of the 3rd Augmented Human International Conference, article No. 27*, 2012.

[24] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Computer Science Department, Stanford University*, 1995.

[25] Nataliya Kosmyna, Franck Tarpin-Bernard, and Bertrand Rivet. Bidirectional feedback in motor imagery bcis: Learn to control a drone within 5 minutes. *Conference on Human Factors in Computing Systems - Proceedings*, 2014.

[26] Nataliya Kosmyna, Franck Tarpin-Bernard, and Bertrand Rivet. Brains, computers, and drones: Think and control! *Interactions*, 22(4):44–47, 2015.

[27] Karl LaFleur, Kaitlin Cassady, Alexander Doud, Kaleb Shades, Eitan Rogin, and Bin He. Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain-computer interface. *Journal of Neural Engineering*, 10(4), 2013.

[28] Robert Leeb, Doron Friedman, Gernot R. Müller-Putz, Reinhold Scherer, Mel Slater, and Gert Pfurtscheller. Self-paced (asynchronous) bci control of a wheelchair in virtual environments: A case study with a tetraplegic. *Computational Intelligence and Neuroscience*, 2007.

[29] Ming-Ai Li, Rui Wang, Dong-Mei Hao, and Jin-Fu Yang. Feature extraction and classification of mental eeg for motor imagery. *5th International Conference on Natural Computation, Icnc 2009*, 1, 2009.

[30] Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, and Bruno Arnaldi. A review of classification algorithms for eeg-based brain-computer interfaces. *Journal of Neural Engineering*, 2007.

[31] Y.-H. Lui, C.-A. Cheng, and H.-P. Huang. Novel feature of the eeg based motor imagery bci system: degree of imagery. 2011.

[32] Xiaoqian Mao, Mengfan Li, Wei Li, Linwei Niu, Genshe Chen, Bin Xian, and Ming Zeng. Review article: Progress in eeg-based brain robot interaction systems. *Computational intelligence and neuroscience*, 2017, 2017.

[33] Jianjun Meng, Shuying Zhang, Angeliki Bekyo, Jaron Olsoe, Bryan Baxter, and Bin He. Noninvasive electroencephalogram based control of a robotic arm for reach and grasp tasks. 2016.

[34] Jonathan Milgram, Mohamed Cheriet, and Robert Sabourin. One against one or one against all: Which one is better for handwritting recognition with svms? *Guy Lorette. Tenth International Workshop on Frontiers in Handwritting Recognition, Oct 2006, La Baule (France), Suvisoft.*, 2006.

[35] Klaus-Robert Müller, Charles W. Anderson, and Gary E. Birch. Linear and nonlinear methods for brain-computer interfaces. *Journal of Neural Engineering*, 2007.

[36] M. Naeem, C. Brunner, R. Leeb, B. Graimann, and G. Pfurtscheller. Seperability of four-class motor imagery data using independent component analysis. 2006.

[37] Chang S. Nam, Yongwoong Jeon, Young-Joo Kim, Insuk Lee, and Kyungkyu Park. Movement imagery-related lateralization of event-related (de)synchronization (erd/ers): Motor-imagery duration effects. *Clinical Neurophysiology*, 122(3), 2011.

[38] The Central Nervous System OpenStax College, Unknown upload data. URL `https://www.boundless.com/biology/textbooks/boundless-biology-textbook/the-nervous-system-35/the-central-nervous-system-202/brain-cerebral-cortex-and-brain-lobes-767-12000/images/lobes-of-the-cerebral-cortex/`.

[39] Md. Mamun or Rashid and Mohiuddin Ahmad. Multiclass motor imagery classification for bci application. *Iwci 2016 - 2016 International Workshop on Computational Intelligence*, 2016.

[40] G. Pfurtscheller, C. Brunner, A. Schlögl, and F.H. Lopes da Silva. Mu rhythm (de)synchronization and eeg single-trial classification of different motor imagery tasks. *Neuroimage*, 31(1), 2006.

[41] Rajesh P.N. Rao. *Brain-Computer Interfacing - An introduction*. Imotions - Biometric Research Platform, 2013.

[42] Sebastian Raschka, Unknown upload data. URL `http://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/`.

[43] N. Rathipriya, S. Deepajothi, and T. Rajendran. Classification of motor imagery ecog signals using support vector machine for brain computer interface. 2013.

[44] Audrey S. Royer, Alexander J. Doud, Minn L. Rose, Bin He, and Fellow IEEE. Eeg control of a virtual helicopter in 3-dimensional space using intelligent con-trol strategies. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(6):581–589, 2010.

[45] David Shier, Jackie Butler, and Ricki Lewis. *Hole's essentials of human anatomy and physiology - eighth edition*. McGraw-Hill, 2003.

[46] Wendi Song, Xiangzhou Wang, Shuhua Zheng, and Yingzi Lin. Mobile robot control by bci based on motor imagery. *Proceedings*, 2, 2014.

[47] H.-I. Suk and S.-W. Lee. Two-layer hidden markov model for multi-class motor imagery classification. 2010.

[48] Carsten E. Thompson and Merete Bakke. Elektroencephalografi og cerebral funktion. 2004.

[49] Juan Tian and Zhaochen Zhang. Research on algorithm for feature extraction and classification of motor imagery eeg signals. 2017.

[50] Unknown. *EEG pocket guide*. Imotions - Biometric Research Platform, 2016.

[51] Unknown, Unknown upload data. URL `http://biomedicalengineering.yolasite.com/neurons.php`.

[52] Cinnamon L. VanPutte, Jennifer L. Regan, and Andrew F. Russo. *Seeley's Anatomy and Physiology - Tenth edition*. McGraw-Hill, 2014.

[53] L. Wang, G. Xu, and J. Wang et al. Application of hilbert-huang transform in the study of motor imagery tasks. 2008.

[54] Danny Wee-Kiat, Soh Ying-Wei, and Sing-Yau Goh. Development of an autonomous bci wheelchair. *IEEE Symposium on Computational Intelligence in Brain Computer Interfaces (cibci)*, 2014.

[55] B. Xu and A. Song. Pattern recognition of motor imagery eeg using wavelet tranform. 2008.

[56] Han Yuan, Alexander Doud, Arvind Gururajan, and Bin He. Cortical imaging of event-related (de)synchronization during online control of brain-computer interface using minimum-norm estimates in frequency domain. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(5):425–431, 2008.

[57] Rui Zhang, Yuanqing Li, and Yongyong Yan. An intelligent wheelchair based on automated navigation and bci techniques. *IEEE Engineering in Medicine and Biology Society Conference proceedings*, 2014.

[58] Rui Zhang, Yuanqing Li, Yongyong Yan, and et al. Control of wheelchair in an indoor environment based on a brain-computer interface and automated navigation. *AH'12 Proceedings of the 3rd Augmented Human International Conference, article No. 27*, 2016.

Appendix

The Appendix contains all the relevant developed programs used during this project and is divided into two sections, namely, programs developed in Matlab and programs developed in Python.

9.1 Programs developed in Matlab

This section of the Appendix contains all of the relevant programs developed in Matlab during the project. Many of the functions have been slightly modified to serve slightly different purposes, but have not been included as they were deemed irrelevant and to avoid redundancy.

9.1.1 Offline data processing script

```
%% Load data
% This script is for offline processesing EEG data recorded by the Emotiv
% system. It is setup for a specific data structure and requires knowgles
% of the data classes.
%
% This script should be run in sections, some sections can be skipped
% without consequenses.
%
%   Created by Jonas Rossing
%   Last modified: 31-05-2017
%
%%%% CLASSES:
% 1 = right hand
% 2 = left hand
% 3 = both hands
% 4 = Feet
% 5 = Tounge
% 6 = Idle

clear all

% Script to load all appropriate paths for data and classifiers
startPath

% Load in data and sequence (the data should be called data,
% and the sequences should be called All_sequences
```

```
load CecilieT456_MI
load sequenceMI_T456
y = All_sequences;

% Extract MI events and set sampling frequency
fs = 128;
dataF = data(6*fs+1:end,:,:);

% Define bandpass filter
cutoff = [4 25]./(fs/2);
[b,a] = butter(4,cutoff);


%% Cut in 1 sec epoch
dataE = [];
yE = [];
for i = 0:floor(size(dataF,1)/fs)-1
    dataE = cat(3,dataE,dataF((fs*i)+1:(fs*i)+fs,:,:));
    yE = [yE,y];
end
clear dataF
clear y
dataF = dataE;
y = yE;


%% Bandpass filter
for j = 1:size(dataF,3)
    for i = 1:size(dataF,2)
        dataFe(:,i,j) = filtfilt(b,a,dataF(:,i,j));
    end
end
clear dataF
dataF = dataFe;

%% Remove DC component
for i = 1:size(dataF,3)
    for i2 = 1:size(dataF,2)
        dataF2(:,i2,i) = detrend(dataF(:,i2,i));
    end
end
clear dataF
dataF = dataF2;

%% Create feature vector

% Set channels
ChannelPlacement =
    string({'O2';'FPz';'P4';'F3';'O1';'F4';'Pz';'Fz';'P3';'C3';'TP10';'Cz';'C4';'TP9'});
clear X
channels = [3 4 6 7 8 9 10 12 13];

X = featureFuncOffline(dataF,fs,channels);

% Assign in struct
classes = cellstr(num2str(y'));
```

```
EEGdata = array2table(X);
EEGdata.Classes = categorical(classes);

%% Test accuracy
load TM_test    % Load classifier
yfit = trainedModel.predictFcn(EEGdata);
yfit = str2double(char(yfit));

y2 = yfit-y';
y2(y2==0)=[];
accuracy = 100-length(y2)/length(y)*100
```

9.1.2 Data recording script

```
%% Initiation
% This is a scipt used for EEG data recording with the Emotiv system.
% The script is meant to be run simultaneously with a series of images
% corresponding to MI movements on another computer
%
%      Requirements:
%            - Lab streaming layer (lsl)
%            - Input from another system, ex Python
%
%      This code should be run in sections
%
%   Created by Jonas Rossing
%   Last modified: 29-05-2017

% set paths
startPath

% Run this section once for each dataset
% Load lsl libary
disp('Loading the library...');
lib = lsl_loadlib();

% Set sampling frequency
fs = 128;

% Counters
k = 1;
k3 = 1;

% Set duration and number of trials in each session
trialDur = 10;
trialStart = 1;
trialsAM = 12;

%% Data recording
% Search for a data stream
disp('Resolving an EEG stream...');
result = {};
```

```
% Pause 5.5 seconds to allow subject to adjust
pause(5.5);

% Standby if no stream is available
while isempty(result)
    result = lsl_resolve_byprop(lib,'type','EEG');
end

% create a new inlet
disp('Opening an inlet...');
inlet = lsl_inlet(result{1});

% Run for a session
for j = trialStart:trialStart+trialsAM-1

    % Counters
    k = 1;
    k2 = 1;
    k4 = 1;
    fprintf('\n');
    disp('Now receiving data...');
    while k2<fs*trialDur+1
        % get data from the inlet
        [vec,ts] = inlet.pull_sample();

        % Save the data
        data(k,:,j) = vec;
        k = k+1;
        k2 = k2+1;
    end

    disp(['Trial ',num2str(j),' done']);
end

fprintf('Done recording\n');

% Set start for new session
trialStart = trialStart+trialsAM;

%% Save the data
save('dataEEG','data');
```

9.1.3 Live feedback script

```
%% Initiation
%
%   This script is for live testing a BCI system online with feedback. It
%   will extract features and classify them according to a specific classifier.
%
%   This scipt requires 2 computers setup for UDP transfer. The second
%   computer will handle user feedback, while this one handles the data
%   processing.
%   A modified version of this scipt exist for 1 computer only setup
```

```
%
%   Created by Jonas Rossing
%   Last modified: 29-05-2017

% Set paths
startPath

% load lsl libary, UDP info and classifier
disp('Loading the library...')
lib = lsl_loadlib();
load TM_6C_SVMC
load udpA

% Set sampling frequency and channels
fs = 128;
%channels = [3 7 8 9 10 13];
channels = [3 4 6 7 8 9 10 12 13];

% Define bandpass filter
cutoff = [4 25]./(fs/2);
[fb,fa] = butter(4,cutoff);

% Search for a data stream
disp('Resolving an EEG stream...')
result = {};

% Standby if no stream is available
while isempty(result)
    result = lsl_resolve_byprop(lib,'type','EEG');
end

% create a new inlet
disp('Opening an inlet...');
inlet = lsl_inlet(result{1});

% Counters
k = 1;
k2 = 1;
k4 = 1;

disp('Now receiving data...');
while true
    %while true
    % get data from the inlet
    [vec,ts] = inlet.pull_sample();

    % Save data
    data(k,:) = vec;
    k = k+1;
    k2 = k2+1;

    % Split into 1s epochs with 1/2s overlap
    if size(data,1) == fs/2
        if  exist('data_old') == 0
            data_old = data;
            clear data
```

```
            k = 1;
        else
            data_new = [data_old;data];
            data_old = data;
            clear data
            k = 1;

            % Filter data
            for i = 1:size(data_new,2)
                dataDT(:,i) = filtfilt(fb,fa,data_new(:,i));
            end
            % Detrend data
            for i2 = 1:size(dataDT,2)
                data_A(:,i2) = detrend(dataDT(:,i2));
            end

            % Extract features
            X = featureFuncOffline_Jonas(data_A,fs,channels);
            EEGdata = array2table(X);
            yfit(k4) = trainedModel.predictFcn(EEGdata);
            yfit(k4)
            k4 = k4+1;

            % Send data to second computer
            fprintf('\n');
            if strcmp(char(yfit(k4-1)),'1')
                fprintf(udpA,'1');
            elseif strcmp(char(yfit(k4-1)),'2')
                fprintf(udpA,'2');
            elseif strcmp(char(yfit(k4-1)),'3')
                fprintf(udpA,'3');
            elseif strcmp(char(yfit(k4-1)),'4')
                fprintf(udpA,'4');
            elseif strcmp(char(yfit(k4-1)),'5')
                fprintf(udpA,'5');
            else
                fprintf(udpA,'6');
            end
        end
    end
end
```

9.1.4 Live simulation script

```
%% Initiation
% This scipt is for live simulation of drone flight with a BCI system.
% It will preprocess and extract features and classify them, while
% simulating a plane in a virtual reality.
%
% The script requires some additions to work
%   - featureFunc.m
%   - vrtest.m
%   - biosig toolbox
```

```
%   - lsl
%   - a classifier
%   - a data stream from a secondary program, ex Python
%
%   Created by Jonas Rossing
%   Last modified: 29-05-2017

% Set paths
startPath

% Load lsl libary and classifier
clear all
disp('Loading the library...')
lib = lsl_loadlib();
load TM_6C_SVMQ

% Set sampling frequency and channels
fs = 128;
channels = [3 4 6 7 8 9 10 12 13];

% Define bandpass filter
cutoff = [4 25]./(fs/2);
[fb,fa] = butter(4,cutoff);

% create and open the vrworld object
w = vrworld('vrtkoff.wrl', 'new');
open(w);

% create the vrfigure showing the virtual scene
fig = vrfigure(w, 'Viewpoint', 'Ride on the Plane');

% get the manipulated airplane node
airpln = vrnode(w, 'Plane');
% read plane initial translation and rotation
originalTranslation = airpln.translation;
originalRotation = airpln.rotation;
originalRotation = [0 0 0 0];
vrdrawnow;

% Predefine location parameters
a = 0;
b = 0;
c = 0;

% Search for a stream
disp('Resolving an EEG stream...')
result = {};
while isempty(result)
    result = lsl_resolve_byprop(lib,'type','EEG');
end

% create a new inlet
disp('Opening an inlet...');
inlet = lsl_inlet(result{1});

k = 1;
```

```
k2 = 1;
k3 = 0;
k4 = 1;
k5 = 1;
disp('Now receiving data...');

while true
    %while true
    % get data from the inlet
    [vec,ts] = inlet.pull_sample();

    % Save data
    data(k,:) = vec;
    k = k+1;
    k2 = k2+1;

    % Split into 1s epochs with 1/2s overlap
    if size(data,1) == fs/2
        if  exist('data_old') == 0;
            data_old = data;
            clear data
            k = 1;
        else
            % Time the script
            toc
            tic
            data_new = [data_old;data];
            data_old = data;
            clear data
            k = 1;

            % Filter data
            for i = 1:size(data_new,2)
                dataDT(:,i) = filtfilt(fb,fa,data_new(:,i));
            end
            % Detrend data
            for i2 = 1:size(dataDT,2)
                data_A(:,i2) = detrend(dataDT(:,i2));
            end

            % Extract features
            X = featureFunc(data_A,fs,channels);
            EEGdata = array2table(X);
            yfit(k4) = trainedModel.predictFcn(EEGdata);
            k4 = k4+1;
        end
    end

    % Calculate new translation and rotation and draw it
    if k4 > 1
        if k5 > 4
            [airpln.translation,airpln.rotation,a,b,c] =
                vrtest(str2double(char(yfit(k4-1))) ...
            ,originalTranslation,originalRotation,a,b,c);
            vrdrawnow;
            k5 = 1;
```

```
            %toc
        else
            k5 = k5+1;
        end
    end
end
```

9.1.5 Feature extraction function

```
function X = featureFunc(data,fs,Ch)
% X = featureFunc(data,fs,Ch) is used for feature extraction for a BCI
% system.
%
%       Requirements:
%           Biosig toolbox
%
%       Input:
%           data: a [datapoint x channels x trials] matrix with EEG data
%           fs: The sampling frequency of the system
%           Ch: A [1 x N] vector with channels corresponding to the EEG
%               data
%
%       Output:
%           X: A [trials x N*numF], where numF is the number of different
%           features.
%
%   Created by Jonas Rossing
%   Last modified: 29-05-2017


% Remove unwanted channels
data2 = data(:,Ch,:);

% Number of features, should be adjusted is features is added or removed
numF = 5;

% Input to AAR
mode = [1,2];
MOP = [5,0];

% Prelocate
X = zeros(size(data2,3),numF*length(Ch));

% Feature extraction
for i = 1:size(data2,3)
    for j = 0:size(data2,2)-1
        X(i,numF*j+1) = log10(bandpower(data2(:,j+1,i), fs, [8 13]));
        AARparameters = mean(aar(data2(:,j+1,i),mode,MOP));
        X(i,numF*j+2) = AARparameters(1,1);
        X(i,numF*j+3) = AARparameters(1,5);
        X(i,numF*j+4) = log10(HjorthPara(data2(:,j+1,i),'com'));
        X(i,numF*j+5) = log10(bandpower(data2(:,j+1,i), fs, [12 16]));
    end
```

```
end
end
```

9.1.6 Hjorth parameter function

```
function [para] = HjorthPara(f,varargin)
% [para] = HjorthPara(f,varargin) calculates 2 of the Hjort parameters
% mobillity and complexity.
%
%       Input:
%           f: a [N x 1] vector of EGG data
%           varagin: A string, either 'mob' or 'com' for which parameter is
%           wanted.
%
%       Output:
%           para: The chosen Hjort parameter
%
%   Created by Jonas Rossing
%   Last modified: 29-05-2017

% Calculate the derivatives
n = length(f);
dxf = diff([0;f]);
ddxf = diff([0;dxf]);
mx2 = mean(f.^2);
mdx2 = mean(dxf.^2);
mddx2 = mean(ddxf.^2);

mob = mdx2/mx2;

if strcmp('mob',varargin) == 1;
    para = sqrt(mob);         % Mobility
elseif strcmp('com',varargin) == 1;
    para = sqrt(mddx2 / mdx2 - mob); % Complixity
else
    fprintf(2,'Wrong argument input');
end

end
```

9.1.7 Zero-crossings function

```
function zerocrossings = zerocross(data)
% zerocrossings = zerocross(data) calculates the zero-crossing of EEG data.
%
%   Input:
%       data: a [N x 1] vector of EEG data
%
%   Output:
%       zerocrossing: The zerocrossing of the data
```

```
%
%   Created by Cecilie Lochet
%   Last modified: 29-05-2017

t = 1:length(data)-1;
zerocrossings = 0 ;
for i = 1:length(t)
    if sign(data(i)) ~= sign(data(i+1))
    zerocrossings = zerocrossings + 1 ;
    end
end
end
```

9.1.8 Spectral entropy function

```
function SpecEnt = SpectralEnt(data)
     x = fftshift(fft(data));
     PSD = abs(x).^2;
     sum = 0;
     for i = 1:length(data)
         sum = sum + PSD(i);
     end
     PSD_N = PSD/sum;

     SpecEnt = 0.0;
     for i = 1:length(data)
         Ent = Ent - PSD_N(i)*log2(PSD_N(i));
     end
end
```

9.1.9 Feature selection function

```
function [acTable, acMax, maxFeatures] = featureSelectionFcn(dataR, channels,
    classes, fs, classifier)

% [acTable, acMax, maxFeatures] = featureSelectionFcn(data, channels, classes,
    fs, classifier)
% is used for finding the best combination of features for EEG based BCI
% for a specific set of classes.
%
%   Requirements:
%               Modified classification script(s)
%               Biosig toolbox
%
%   Input:
%       data: EEG data with dimensions [data x channels x trials] (a trial
%       correspond to 1 class)
%
%       channels: A [1 x N] vector with channels corresponding to the EEG
%       data
```

```
%
%       classes: A [1 x trials] dimension vector containing the true
%       classes
%
%       fs: The sampling frequency
%
%       classifier: Specifies which classifier to use. The options are SVMQ
%       (SVM Quardratic) and SSD (subspace discriminant)
%
%
%   Output:
%        acTable: Table with numeric indication of features and their
%        corresponding accuracy. Look in the script for detail on each
%        feature
%
%        acMax: Max accuracy optained
%
%        maxFeatures: A vector containing the features corresponding the
%        the max accuracy optained.
%
%   Feature translation table: (Update as necessary)
%        1: Bandpower (8-13Hz)
%        2: Hjorts activity (varians)
%        3: Hjorts mobility
%        4: Entropy (log energy)
%        5: Entropy (Shannons)
%        6: Hjorts complexity
%        7: Bandpower (12-16Hz)
%        8: Bandpower (Total))
%        9-13: AAR (Adaptive auto-regressive model)
%        14: Spectral entropy
%        15: Zero-crossing
%        16: Bandpower (4-8Hz)
%
%   Note:
%        This script can take a very long time to run, consider letting it
%        run through the night if it have to evaluate a lot of features.
%        The accuracy is around +-8% depending on classification method,
%        but this can be decreased by increasing the number of classifications.
%
%   Created by Jonas Rossing,
%   Last edited 04-05-2017

% Set path to classifiers
startPath

% Start tic count
tic
fprintf('\nInitializing')

% Select only wanted channels
data = dataR(:,channels,:);
y = classes;

% Predefine counters
k2 = 0;
```

```
k3 = 1;
mode = [1,2];
MOP = [5,0];

% Select number of features (Adjust this before using the script)
numF = 16;

% Calculated the total number of iterations needed
NI = 0;
for p = numF:-1:1
    NI = NI + factorial(numF)/(factorial(p)*factorial(numF-p));
end


% Prelocate
pickOld = zeros(NI,numF);
for i3 = 1:numF
    VaNames(i3) = {['FeatureColumn',num2str(i3)]};
end


% Run for all lengths of combinations
for p = numF:-1:1
    CCount = 0;
C = factorial(numF)/(factorial(p)*factorial(numF-p));

    % Run till all combinations have been found
    while CCount < C

        % Select random feature combination and check if already exists
        pick = sort(randperm(numF,p));
        if length(pick) < numF
            pick(numF) = 0;
        end
        [~,index] = ismember(pickOld,pick,'rows');
        if any(index==1) == 0

            % Create feature vector (Insert or delete features as required,
            % just remember jo adjust numF accordingly)
            for i = 1:size(data,3)
                k = 1;
                for j = 0:size(data,2)-1

                    AARparameters = mean(aar(data(:,j+1,i),mode,MOP));

                    if any(pick==1) == 1
                        X(i,k) = log10(bandpower(data(:,j+1,i), fs, [8 13]));
                        k = k +1;
                    end
                    if any(pick==2) == 1
                        X(i,k) = log10(var(data(:,j+1,i)));
                        k = k+1;
                    end
                    if any(pick==3) == 1
                        X(i,k) = log10(HjorthPara(data(:,j+1,i),'mob'));
                        k = k+1;
```

```
            end
            if any(pick==4) == 1
                X(i,k) = log10(abs(wentropy(data(:,j+1,i),'log
                    energy')));
                k = k+1;
            end
            if any(pick==5) == 1
                X(i,k) = log10(abs(wentropy(data(:,j+1,i),'shannon')));
                k = k+1;
            end
            if any(pick==6) == 1
                X(i,k) = log10(HjorthPara(data(:,j+1,i),'com'));
                k = k+1;
            end
            if any(pick==7) == 1
                X(i,k) = log10(bandpower(data(:,j+1,i), fs, [12 16]));
                k = k+1;
            end
            if any(pick==8) == 1
                X(i,k) = log10(bandpower(data(:,j+1,i), fs, [4 25]));
                k = k+1;
            end
            if any(pick==9) == 1
                X(i,k) = AARparameters(1,1);
                k = k+1;
            end
            if any(pick==10) == 1
                X(i,k) = AARparameters(1,2);
                k = k+1;
            end
            if any(pick==11) == 1
                X(i,k) = AARparameters(1,3);
                k = k+1;
            end
            if any(pick==12) == 1
                X(i,k) = AARparameters(1,4);
                k = k+1;
            end
            if any(pick==13) == 1
                X(i,k) = AARparameters(1,5);
                k = k+1;
            end
            if any(pick==14) == 1
                X(i,k) = specEntropy(data(:,j+1,i));
                k = k+1;
            end
            if any(pick==15) == 1
                X(i,k) = zerocross(data(:,j+1,i));
                k = k+1;
            end
            if any(pick==16) == 1
                X(i,k) = log10(bandpower(data(:,j+1,i), fs, [4 8]));
                k = k+1;
            end
        end
    end
```

```
            clear AARparameters

            % Save the feature selection in pickOld
            k2 = k2+1;
            for i2 = 1:length(pick)
            pickOld(k2,i2) = pick(i2);
            end
            CCount = CCount+1;

    % Classify data
    % Run through classifier 5 times and mean (Improves true accuracy, but
    % greatly increases computational time)
    % Replace or remove classifiers as necessary
    clear EEGdata
    clear acT
    for MK = 1:5
        classes = cellstr(num2str(y'));
        EEGdata = array2table(X);
        EEGdata.Classes = categorical(classes);
        if strcmp(classifier,'SVMQ')
            [~,acT(MK)] = SVMQ_model(EEGdata);
        elseif strcmp(classifier,'SSD')
            [~,acT(MK)] = SSD_model(EEGdata);
        elseif strcmp(classifier,'SVMC')
            [~,acT(MK)] = SVMC_model(EEGdata);
        else
            [~,acT(MK)] = SVML_model(EEGdata);
        end
    end

    acSTD(k3) = std(acT);
    ac(k3) = mean(acT);
    k3 = k3+1;
    fprintf('\n')
    disp(['Iteration: ',num2str(k3-1),' of ',num2str(NI),' Done'])

    clear X

        else
        end
    end
end

    % Create table with features corresponding to accuracy
    acTable = array2table(pickOld,'VariableNames',VaNames);
    acTable.Accuracy = cellstr(num2str(ac'*100));
    acTable.std = cellstr(num2str(acSTD'*100));
    acMax = max(ac);
    maxFeatures = pickOld(ac==acMax,:);

    fprintf('\nAccuracy determination completed\n')
    toc
    fprintf('\nMax accuracy attained was: acMax = ')
    disp([num2str(acMax*100),'%']);
end
```

9.1.10 UPA setup

```
%% UDP
% setup script for communication between computer on the same network
%   Created by Jonas Rossing
%   Last modified: 29-05-2017

%Set IP and port
ipA = '10.16.175.30'; portA = 1335; % IP and port for computer 1
ipB = '10.16.163.131'; portB = 1336; % IP and port for computer 2
%% Create UDP Object
udpA = udp(ipB,portB,'LocalPort',portA);
%% Connect to UDP Object
fopen(udpA)
```

9.2 Programs developed in Python

This section of the Appendix contains the program borrowed from Per Bækgaard, DTU compute, for the data acquisition process. The program has been heavily modified to suit the necessities of the project.

```
# In[1]:

# Emotiv data recording code, can be found on Github
# The code itself is heavily modified to allow live recording
#
#   Requirements:
#              Lab Streaming Layer (LSL)
#              emokit (can be obtained through Github or with pip install)
#              For other requirements, see EmotivMaster on Github
#
#   Last modified by Jonas Rossing, 31-05-2017


    # Import packes
import time
import sys
import ctypes
import os
from ctypes import *
from numpy import *
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import threading
import datetime
import numpy
import scipy.fftpack
import scipy
import collections
import numpy
```

```
### load emokit python package
sys.path.append('Set path')
from emokit.emotiv import Emotiv


from pylsl import StreamInfo, StreamOutlet

# Create stream info
info = StreamInfo('Emotive', 'EEG', 14, 128, 'float32', 'myuid34234')

# Create outlet
outlet = StreamOutlet(info)


#define all electrodes
all_electrodes = ['AF3','AF4','F3','F4', 'F7', 'F8', 'FC5', 'FC6', 'T7','T8',
    'P7', 'P8', 'O2', 'O1']
electrodes = all_electrodes

# In[2]:

    # set run time
tt = 10000
eeg = collections.OrderedDict()
for i in all_electrodes:
    eeg[i] = []

    # Pre define matrix, because Python
data = numpy.zeros(shape=(14,128*tt))
def column(matrix, i):
    return [row[i] for row in matrix]

# In[]

    # Couter
    j = 0

    print ('foo starts at ', str(datetime.datetime.now()))

    sf = 128.0 #sampling frequency
    c = 0 #initalize c, c is used to count data points taken
    stop = 0 #intialize stop, used to stop emokit decoding
    rva = 1.0/sf+1.0/sf*0.1-1.0/sf*0.07 # Pause between samples, different
                                        # from PC to PC


    validate_timer = 0
    # set it to 1,if we want to check that the code is excecuted at the
    # but it will crash correct frequency,

    if __name__ == "__main__":
        with Emotiv(display_output=False, verbose=True) as headset:
            while stop == 0:
                packet = headset.dequeue()
                next_call = time.time()
```

```
            starttime = time.time()

            if packet is not None:
                print ('starting..........')
                while c < tt * sf:

                    # load eeg data into dicitonary
                    for k in electrodes:
                        eeg[k].append(packet.sensors[k]['value'])

                    # Assign data in matrix for output, not optimal but
                    # works
                    data[0][j] = eeg['AF3'][j]
                    data[1][j] = eeg['AF4'][j]
                    data[2][j] = eeg['F3'][j]
                    data[3][j] = eeg['F4'][j]
                    data[4][j] = eeg['F7'][j]
                    data[5][j] = eeg['F8'][j]
                    data[6][j] = eeg['FC5'][j]
                    data[7][j] = eeg['FC6'][j]
                    data[8][j] = eeg['T7'][j]
                    data[9][j] = eeg['T8'][j]
                    data[10][j] = eeg['P7'][j]
                    data[11][j] = eeg['P8'][j]
                    data[12][j] = eeg['O2'][j]
                    data[13][j] = eeg['O1'][j]

                    dataSend = column(data,j)
                    j = j+1

                    # Send data to matlab
                    outlet.push_sample(dataSend)

                    c = c + 1
                    #print ('c', c)

                    if c == tt * sf:
                        print (c)
                    #this is for debugging
                    if validate_timer == 1:
                        print(datetime.datetime.now())
                    #sleep till the period ends
                    time.sleep(rva)
                stop = 1
                endtime = time.time()


        print('total runtime', endtime - starttime)
```