# Designing a BCI system: Preprocessing, feature selection and classification of EEG for the navigation of a mobile robot

*MSc Dissertation*

Hinrik Jósafat Atlason

Department of Computer Science
University of Essex

16th September 2005


*This report is the dissertation of a research project that was undertaken for the award of a MSc qualification in Computer Science - Artificial Intelligence.*

Title: Designing a BCI system: Preprocessing, feature selection and classification of EEG for the navigation of a mobile robot.
Student: Hinrik Jósafat Atlason.
Supervisor: Dr. Francisco Sepulveda
16th September 2005

**Acknowledgements**

The author would like to thank his supervisor, Dr. Francisco Sepulveda, for his support and effort, Mr. Robin Dowling for the LEDs, and the students that were present in the Brooker laboratory during experiments for their patience.

Hinrik Jósafat Atlason

**Abstract**

*This report documents a research conducted into the design of brain-computer interface systems. It describes relevant aspects of neuroscience, signal processing and artificial intelligence. It implements and describes a system that uses recorded electroencephalogram from a persons scalp and filters it, selects and extracts features, and lastly, uses it to train three different classifiers, one linear and two non-linear. Feature selection is based on the class overlap of the signals that relate to the features and is measured using Davis-Bouldin indices. The predictions of the classifiers can be used for the navigation of a mobile robot in up to four different directions. Results are varying but interesting questions are rased. Professional research practice is omnipresent in most design choices for which statistical or theoretical reasons are given. It proposes an outline of a design for a practical brain-computer interface system based on the selected features and techniques.*

Keywords*: Brain-computer interface systems, electroencephalogram, neuroscience, signal preprocessing, feature selection and artificial neural networks.*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

For many years researchers have been intrigued by the biological brain, its structure and functionality as well as its immense computational power, a result of the interaction of it's basic elements, the neurons. Although the brain has been a research topic for hundreds of years, it was not until the early 1940 that researchers [40] were able to devise a unit which conformed with the logical architecture of the biological neuron. This artificial neuron was capable of performing elementary logical operations on its inputs and determining the truth value of any binary logical operation when connected in a temporal sequence, thus resembling its biological counterpart - the neural network. [2] The artificial neurons could either be emitting a signal or not, i.e., on or off. This is also true for biological neurons which output spikes of electrical impulses which become detectable on a persons scalp as a result of the sheer number of neurons that are active at any given time. Each individual spike is rather uninformative of the state of the whole brain but the sequence and timing of a collection of these are possible indicators of a persons intentions. The rhythms, commonly referred to as *electroencephalogram* (EEG), that are produced by the spikes have been used by software which is designed to detect the intention of the user and apply this information to a practical problem such as controlling a cursor on a computer screen [36] or a mobile robot [31]. Both have a great practical application for individuals that suffer from severe cases of amyotrophic lateral sclerosis, brainstem stroke or injury to brain or spinal cord and are in a locked-in state where they have no voluntary control over their muscles.

These *brain-computer interface* (BCI) systems are the focus of this report which describes research into BCI systems in general as well as their function specific parts which are signal processing, feature

selection and extraction, and classification.

### Problem statement and project type

EEG can be considered as stochastic processes with little or no correlation between their shape and the brain function that they encode. Statistical methods must therefore be applied to find descriptive features in the EEG which can be used for the training of classifiers that will learn to map EEG components to specific brain functions. This is what the bulk of this project is about, i.e., a research into the design of a system that can extract EEG components and used them to predict different intentions. This involves thorough research into statistical feature selection and classification techniques, as well as signal acquisition and processing.

### Goals

The desired outcome of this project was, firstly, a better understanding of proper research conduct, techniques, documentation and, secondly, to develop further competence in applying recognised signal processing and artificial intelligence techniques to a practical problem and draw valid conclusions from the results and experience.

What the practical aspects are concerned, the goal was to create a design outline of an application which could be implemented to produce a practical real-time BCI system which could control a mobile robot in one of up to four directions based on predictions of user intentions using selected EEG features. Each part of the system should be chosen from a number of candidates on the basis of a statistical measure.

### Project overview

This project included theories from a number of different fields such as neuroscience, biological sciences, anatomy, calculus, linear algebra, boolean logic, signal processing, algorithms, machine learning, programming, software engineering and research practice & methodology. Some of these to more extent than others such as theories from neuroscience and signal processing which the author was not familiar with prior to the project. These were therefore given priority during literature reviews and background

research as well as some emphasis in the documentation.

### Relevant courses and materials beyond curriculum

Two terms of taught courses preceded this project and some of them provided valuable knowledge in related areas as well as an opportunity to apply standard techniques to a practical problem. These courses are listed below, accompanied by a short summary of the topics which they covered that were relevant to this project.

- $CC482 - G - AU$ *Machine learning and data mining (Dr. P. Scott)*:
  - Strength and limitations of machine learning techniques.
  - Application of machine learning techniques.
  - Evaluation of learning procedures.
  - Practical applications of machine learning techniques.
  - Supervised/unsupervised learning.
  - Neuron models.
  - Activation functions.
  - Feed-forward neural networks (perceptron, RBF networks).
  - Error-backpropagation algorithm.
  - The delta learning rule.
  - Gradient descent methods.
  - Competitive learning.
  - Clustering (Kohonen nets).
  - Overfitting.
- $CC461 - G - SP$ *Artificial neural networks (Dr. J. Gan)*:
  - Practical applications of artificial neural networks.
  - Supervised/unsupervised learning.
  - Neuron models.
  - Activation functions.
  - Feed-forward neural networks (perceptron, RBF networks, SOM networks).
  - Learning rules.
  - Clustering (SOM, K-NN).
- $CC402 - G - SP$ Professional practice & research methodology (Professor R. Turner):

  - Experimental procedures
  - *CC*401 project guidelines.
  - Research methodology.

- *CC*483 – *G* – *AU Evolutionary computation (Dr. Q. Zhang)*:

  This module include workshops where MATLAB was used to program various algorithms. This proved to be very useful in this project since most of the code was written in MATLAB.

- *CC*468 – *G* – *SP Fuzzy logic and hybrid systems (Dr. H. Hagras)*:

  This module also included workshops where robot controllers were programmed in C/C++ which was also the case in this project.

This project required extensive background knowledge of other topics that went beyond the MSc curriculum. Most of them were related to neuroscience and signal processing for which individual investigations needed to be conducted. These theories were omnipresent throughout the project and constituted for more than 3/4 of the total background research. Becoming familiar with these topics required significant effort and determination as well as much experimentation. Some of these were not included in the report for reasons established with experimentation and background research. The following is a summary of the topics.

- *Neuroscience* :
  - Biological neurons.
  - Biological neural networks.
  - Motor cortex.
  - Primary somatosensory cortex.
  - Association cortex (Somatosensory, visual, auditory).
  - Neural pathways to muscles.
  - EEG.
  - Brain rhythms.

- *Signal processing* :
  - Feature domains (spatial, frequency, time and time-frequency).
  - Principal component analysis.
  - Independent component analysis.

- Filtering.
- Wavelets.
- Fourier transform.
- Spectral density of signals.
- Autoregressive models.
- Digital to analog conversion (for different types of filters).
- Signal re-sampling.
- Laplace probability density.
- Learning vector quantisation networks.
- Probabilistic neural networks.

### Related work

BCI systems have been subject to extensive research and improvements, most notably by a group from the Department of Medical Informatics of Graz University of Technology in Austria lead by G. Pfurtscheller. Their research involves, among others, the classification of motor imagery related EEG [55] and [53], using EEG produced by the event-related desynchronization and event-related synchronization phenomena for classification [52] and classification of movement related EEG [46]. They document having used learning vector quantisation neural networks [57], while others have also tried Multiple monotonic neural network [70] and linear classifiers [74] and [4] which performed surprisingly well considering the non-linear nature of the EEG.

The applicability of BCI system in practice has also been studied. One paper is of special interest since it is closely related to this project as it involves EEG-based BCI systems designed to control a mobile robot. C. Kyoung ho and M. Sasaki [31] describe a system that can learn to control a mobile robot when trained with EEG and user feedback.

Other related research include the work of C. Guger *et al.* [25], M. Middendorf *et al.* [42], G. E. Birch *et al.* at the Neil Squire Foundation [5], B. Obermaier *et al.* [47], W. D. Penny *et al.* [49], B. O. Peters *et al.* [51] and E. J. X. Costa and E. F. Cabral Jr. [13].

### Report structure

The report begins by giving a general overview of the problem domain in chapter 2 where different aspects of a typical BCI system are described. The methodology that was adopted is described in chapter

3, and the results obtained by applying the respective techniques are given in chapter 4. Chapter 5 contains a discussion on the choices made in 3 and the results in 4 as well as general considerations on the project. Project management issues are discussed in section 6, and the report is concluded in chapters 7 and 8 where future extensions are discussed.

Readers are advised to make use of the summary of abbreviations, notation and symbols provided in appendix A whilst reading this report.

# Chapter 2

# Background

*This chapter introduces the theories that underlie BCI systems development. It touches upon topics such as neuroscience, EEG, signal preprocessing, filtering, feature selection and classification.*

The most important aspect of BCI systems and classification systems in general, on all but the simplest data sets, is being familiar with the data, i.e., understanding its features and how it is obtained. Without this knowledge, one may obtain reasonable results but there is no guarantee that the same method will prove effective for a different data set belonging to the same population, e.g., with a different test subject. The reason for this is that data sets may posses features which are special for the particular set and contain idiosyncrasies which are not reflected in the whole population. It is, therefore, pivotal that the data source and the structure of the data is examined closely to ensure that the classification is performed on the data which best describes features of the whole population.

Following sections are devoted to this important part of any BCI system. Starting with the brain and its structure, from which the data will ultimately be extracted from, followed by a discussion on different types of signals which can be obtained from the brain. The remainder of the chapter has been designated to other aspects of the project such as the experiments, filtering, feature selection and extraction and, lastly, classification.

## 2.1 The brain and nervous system

The human brain, its structure and functionality, has intrigued researchers for many years and although extensive theoretical knowledge has been accumulated in this area, [33] [32] [24] [43], it is still subject to speculations regarding its basic elements - the neurons - and how certain neuronal firing patterns can result in a unique perception of, say, ones surroundings.

The signals that were used in the system were directly obtained from the scalps of test subjects. These signals were then processed and classifiers trained to classify them according to the test subjects intentions. The design of the system was heavily reliant on knowledge of the human brain. For example, the signals which are used to train the classifiers in BCI systems are often produced by visualising motor activity and the part of the brain which is of interest is thus the area of the brain which controls these functions. If one did not know that the brain is highly conductive and that activity from a confined area of the brain can in fact be detected throughout the brain, one might discard information obtained from a large part of the brain if the respective area were not directly related to motor activity. This might reduce the performance of the system as valuable information might be lost. The remainder of this section will, therefore, describe some relevant aspects of the brain's structure on a neuronal and global level which served as a foundation for some considerations which arose during the design and implementation phases of the system.

### 2.1.1 Neural networks

Although the body has many different types of neurons, this report will only consider the types of neurons which are associated with the brain. The brain has a number of functionally and structurally distinct neurons but a detailed description of each of them is beyond the focus of this paper.[1] The neurons are the building blocks for neural networks such as the brain. They have limited functionality by themselves but are a part of complicated system capable of extraordinary achievements, or with the words of A. J. Amit [2]:

> ...complex function must be a result of the interaction of large numbers of simple elements...

[1] Excellent texts on this subject include [9], [34], [8] and [38]

The basic elements of neural network are the neurons and the synapses between them. The system under consideration (neural networks) can be divided into three parts which are input, central processing and output. [1] These are in fact also the properties of each individual neuron where they are referred to as *the dendritic arbor*, *soma* and *axon*, respectively. [2] Each neuron in the network can have synapses[2] connected to a number of other neurons which increases the complexity of the network and thus the difficulty of modelling its behaviour. It is hard to conceive the immense complexity of a neural network considering that the brain has about $10^{11}$ neurons, each of which has about $10^4$ synaptic inputs from other neurons. [2]

The inputs to a motor neuron[3] ,depicted in figure 2.1, are often referred to as *spikes* which are basically outputs from other neurons that share a synapse with that neuron. A spike is an *action potential* (AP), i.e., a certain voltage produced by a neuron and carried along its axon, typically of the order of tens of millivolts. [38] [8] [2] Potentials may be reduced as they travel between two neurons, i.e., the *pre-synaptic potential* may be less than the *post-synaptic potential* (PSP). The reason for this is that synapse between two neurons are strengthened if both neurons fire in a short period of time and weakened otherwise. This allows the neural network, and thus the brain, to learn. [8] The higher the PSP, the greater the probability of the receiving neuron firing. [2]

A simplified neuron is depicted in figure 2.1. The neuron receives $N$ inputs from $N$ other neurons. The potential travels along the neurons dendritic arbor with an amplitude determined by the synaptic strength between the transmitting neuron and the receiving neuron. A value $v$ is calculated, e.g., the linear sum of the amplitudes of the inputs, after which the neurons output is determined by the output of the function $\varphi(v)$

### 2.1.2 The cortex

The outer layer of the brain is the *cerebral cortex*. [38] [8] This layer is also referred to as the *neocortex* which is related to the fact that this part of the brain is evolutions latest achievement in this area. [20] [38]

[2]Although, the synapses are actually the point where two axons meet, they shall be used to describe the connection between two neurons in the remainder of this report.

[3]Motor neurons are neurons that aid to produce motor action such as limb movement and the only type of neurons considered in this report.

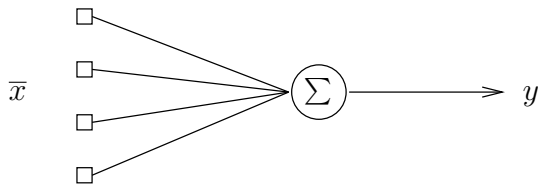



Figure 2.1: The structure of a neuron with input $\overline{x}$ and output $y$.

The preceding discussion on neurons makes assumptions regarding the generality of their structure. This is true in the sense that two neurons in different areas of the brain may have the same logical structure while being used for two distinct purposes. For example, auditory and visually related neurons can have a similar structure although they are positioned in different areas of the brain. [38] [8] Considering earlier BCI research, e.g. [57], where test subject have been asked to imagine movement of their limbs as dictated by visual or auditory cues,[4] one can assume this indicates that there are especially three areas of the brain interesting to this kind of research which include the *motor cortex* which is responsible for motor activity, the *visual cortex* which processes optical signal and the *auditory cortex* which contains neurons dedicated to processing acoustic information. [38] [8] These areas are depicted in figure 2.2 and described in more detail in the following table:

- The motor cortex is located in the back of the *frontal lobe*. The motor cortex has three parts which are indexed with respect to figure 2.2:

  Area 1 is the *primary motor area* (PMA) which contains neurons that control muscle activity. [8]

  Area 2 is the *supplementary motor area* which orders movements in a time sequence. It is related to voluntary and repeated movements. [38]

  Area 3 is the *premotor area* which contains neurons that are used in planning movements. [38]

- Area 4 is the *primary visual cortex* which translates signals that are transmitted from the visual system into signals that, for example, give us a perception of our surroundings. [8]

- Area 5 is the *primary auditory cortex* and is mostly hidden from view and receives most of its inputs from the auditory system. [8]

[4] this will be discussed in greater detail in section 2.2

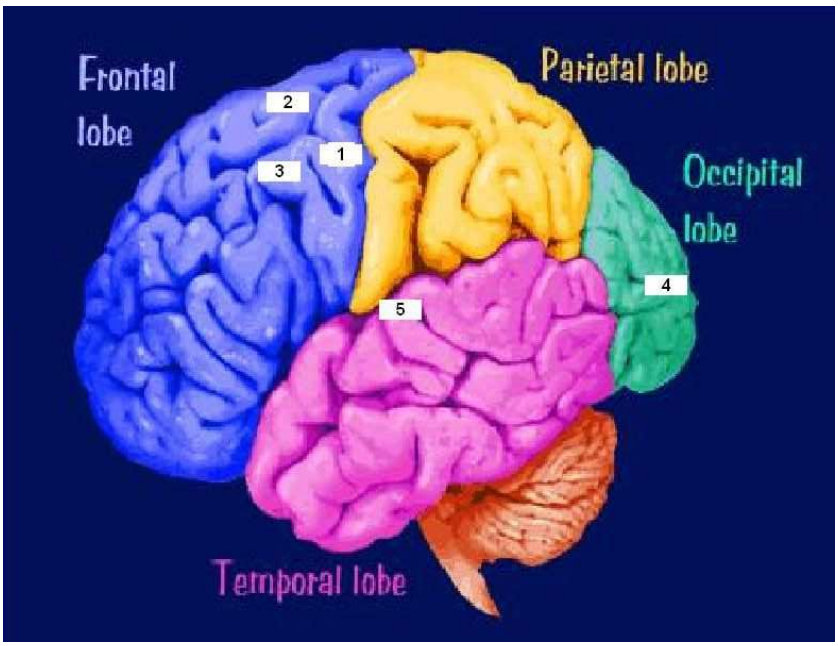



Figure 2.2: The cerebral cortex. Modified from [44].

Areas 2 and 3 receive their input from the parietal lobe where they get a perception of the persons environment as well as the position of her limbs. These areas are both known to be responsible for the planning of voluntary movement and especially complex sequences of movements. Signals that are transmitted from the frontal lobe and the parietal lobe, see figure 2.2, and encode *what* actions are desired to areas 2 and 3 are transformed in area 1 into signals that encode *how* they are to performed. [38]

Area 1 is especially interesting in the context of BCI systems since the areas that control different body parts are readily located on the scalp. This is better described in figure 2.3.

### 2.1.3 Brain rhythms

The brain engages in a number of different activities during the course of a single day, many of which take place without our awareness and many are a direct consequence of our intentions such as motor functions - omitting the philosophical discussion of our intentions as an activity. Some of these can be carried out with seemingly little effort, such as sleep, while others require immediate response, such as re-gaining balance after stumbling. The difference in the "intensity" of brain activities is most evident

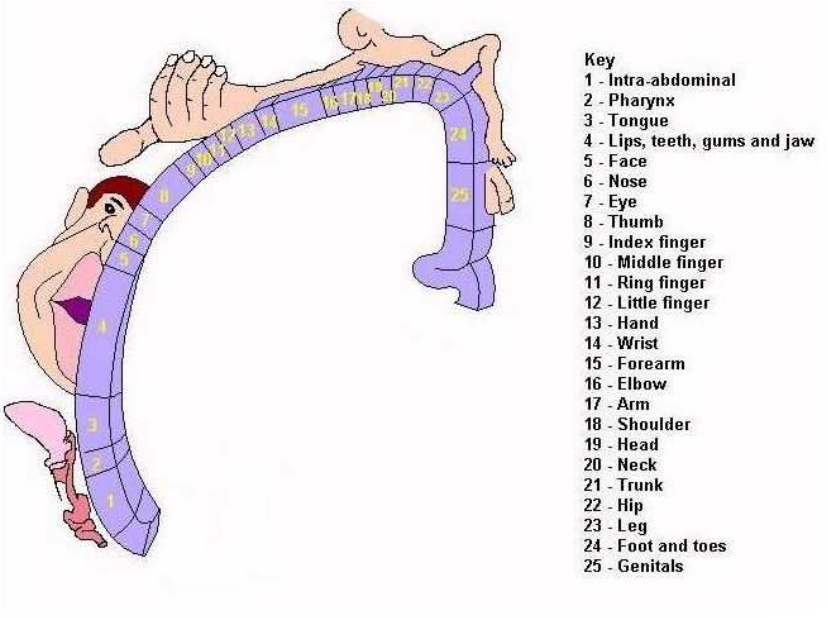



Figure 2.3: The structure of the primary motor area presented as a somatotopic map. From [48].

| Rhythm | Frequency (Hz) | Amplitude | Activity |
|---|---|---|---|
| $\delta$ | $0.5 - 3.5$ | High | Deep sleep |
| $\theta$ | 3.5 - 8 | Varying | Light sleep |
| $\alpha$ | 8 - 13 | High | Relaxation |
| $\beta$ | $13 - 22$ | Low | Concentration |
| $\gamma$ | $22 - 45$ | Varying | Binding (creating perception) |

Table 2.1: The frequencies, amplitudes and activities that are related to the $\alpha$, $\beta$, $\delta$, $\theta$ and $\gamma$ rhythms.

in the *brain rhythms* which are caused by the firing of the brains neurons and are summarised in the below which was adopted from [38] and [8].

An additional brain rhythm is the $\mu$ rhythm which lives in the upper $\alpha$ band. In fact, $\mu$ rhythms are $\alpha$ rhythms confined to the motor cortex.

### 2.1.4 Electroencephalography

An obvious property of any BCI system that uses novel data is that the brain rhythms must be recorded from the brain. There are different ways of doing this, some more time consuming than others.

Recording techniques are commonly divided into two groups; invasive and non-invasive. Invasive techniques such as *positron emission tomography*, and have been used for BCI systems [71] [3], are very costly and time consuming. Non-invasive methods (the obvious choice for this paper) such as recording brain activity with an electroencephalograph which records the electrocortical activity referred to as *electroencephalogram* (EEG), and includes placing electrodes on the scalp of the subject, are much cheaper and have been used to produce satisfactory results. A few of these include the papers by Pfurtscheller *et al* [57], Babiloni *et al* [4], Ramoser *et al* [62] and Cincotti *et al* [10].

EEG are often related to particular behaviours. [38] The EEG that can be sensed on a persons scalp is the interplay of several neurons firing and most apparent in the electrode that is closest to the respective area of the cortex.[5] The EEG that can be sensed on the scalp gives an idea on how synchronised the neural firings are, i.e., the signal will be strong if they fire at the same time and weak if they fire irregularly. [38] Another reason for the satisfactory applicability of EEG in BCI systems is that most of the conscious activities that we engage in are formulated in the cortex[6] and thus close to the electrodes. Using enough electrodes will allow for the recording and localisation, to some extent, of EEG that are related to certain activities.

## 2.2 Experiment

There are two approaches to signal acquisition for BCI systems; using new signals obtained through experiments or using existing, e.g., the *Graz* data sets. The former is more attractive in this case since new, exciting exercises can be specified for the test subjects to do during the experiment. The imagination of limb movement is widely adopted and the EEG that are obtained in this way are referred to as *spontaneous*. Another approach which measures *evoked* signals, i.e., signals that are produced by a change in the persons environment such as a blinking light, is the *P*300 [22] [19] method. The P300 method requires the subject to look at a screen where objects, e.g., boxes or circles, located in the middle of each of the four edges of a computer screen blink at different rates. The EEG in the horn of the *Occipital* lobe of the cortex, figure 2.2, can then be recorded and if the subject looks at, say, the

[5] Although the brain is highly conductive and most EEG can be detected to some extent in any part of the brain. [38]

[6] Specialised systems such as the neural pathways that carry signals from the eyes to the primary visual cortex are known to filter and preprocess the signals along the way. [11]

left most object, then the EEG will contain fluctuations which correspond to the blinking rate of that light which would, say, move a mouse cursor to the left. One drawback of the P300 method is that it requires the subject to have control of her eyes since it is totaly dependent on the gaze direction.

Spontaneous methods do not require any muscle control and thus very well suited for people that suffer from total paralysis. A crucial design factor in such an experiment is how to communicate the nature of the exercise to the test subject, e.g., should she imagine moving her left or right arm. One immediately obvious approach is to use auditory cues such as a recording of the spoken instructions. This, however, introduces some complications such as the possible variation in the length of different instructions as well as the contamination of the EEG caused by the activity in the primary auditory cortex. This contamination is caused by the fact that, in the case of the spoken word, no one instance of the utterance of a word gives much information on the meaning of the word. Rather, audio is dependent on series of instances which will engage the auditory system for a period of time which interferes with EEG over the whole cerebral cortex. Visual cues, such as a blinking light or an arrow on a computer screen, will cause less interference of this kind because of the nature of the visual system which interprets still pictures of what we see. [11]

Timing is a crucial factor in experiments of this kind since the brain is capable of immense parallel processing [2] and thus able to process much information in a relatively short period of time. The brains response time must, therefore, be established and taken into account when designing a protocol for the experiments.

EEG are very weak compared to the potentials produced by muscle activity which are approximately 1000 times stronger. Even eye movements or blinking can produce a potential powerful enough to completely submerge the EEG in noise. Hence it becomes crucial that the test subject is as relaxed as possible during the experiment.

## 2.3 Filtering signals

EEG can contain a lot of noise like the above discussion indicates. This, as well as interference from the power mains, makes the need for proper filtering apparent for a BCI system of this sort. The equipment, described in section 3.1, amplifies and filters the signals during recording. This reduces some of the noise but additional filtering is still required to isolate the various frequency bands. A

brief description is given in the below.

### 2.3.1 Electronic filters

Filters for signal processing come in many different flavors such as digital or analog, linear or non-linear. [69] The ones that are considered in this paper are analog and linear, i.e., they operate on continuously variable signals and they apply linear transformation to the signals. One type of analog filters performs a transform in the Z-domain which is a complex frequency domain representation of the discrete time domain signal. [60] Their operation is described as ...

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_M z^{-M}} \tag{2.1}$$

where $N$ is the input size and $M$ is the filter order. One such is the *Butterworth* filter. It's designed in such a way that the frequency response is maximally flat, i.e., minimal loss of information occurs where in the frequency band the filter's instructed to preserve. [60] A consequence of this is that the Butterworth filter requires a higher order to implement the same *stopband* than, say, a *Chebyshev* filter. This is better described in figure 2.4.

Figure 2.4 illustrates the comparison of two filters, Butterworth and Chebyshev, of orders 1, 2, 4, 6 and 10 as they perform bandpass filtering with a passband between 0.5Hz and 45Hz which is identical to highpass filtering at 0.5Hz and lowpass filtering at 45Hz. One can notice that the frequency response is smoother in the passband of the Butterworth filter which results in minimal loss of valuable information. The figure also shows that less information remains in the *cutoff* frequencies when the order is high.

Any number of frequency bands can be isolated with electronic filters. This is very useful when, say, comparing the classification accuracy of a system with signals from a specific frequency band or a combination of these.

Noise in EEG can also be caused by muscle activity. The blink of an eye or neck movements distinguish them selves from neural activity by being stronger in the order of thousands. However, blinking is so infrequent relative to a reasonable sample rate that their effect can be averaged out.

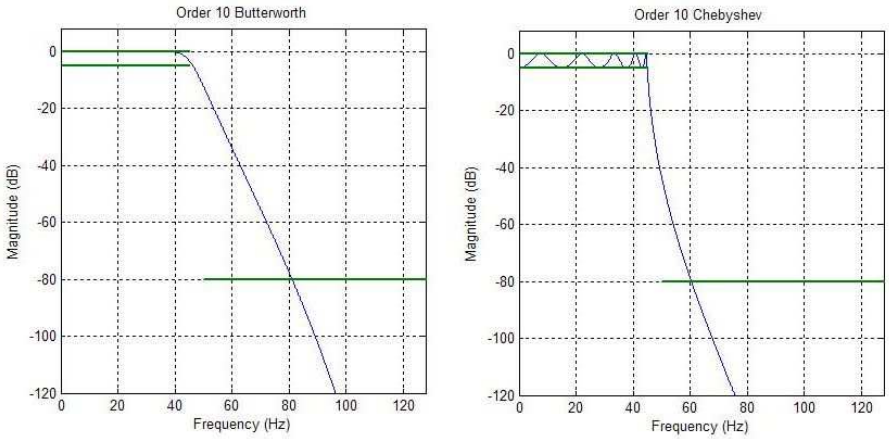



Figure 2.4: The comparison between the frequency response of two filters, the (a) Butterworth and (b) Chebyshev of order 10.

## 2.4 Feature selection

There are many issues and relationships between these that need to be considered when designing a BCI system such as signal acquisition, experimental procedure, filtering, feature selection and classification. The experimental procedure, for example, may affect the choice of features. Examples of this are the $P300$ which engages the primary visual cortex and the motor imagery that engage the motor cortex and parts of the frontal lobe. Feature selection affects the classification accuracy since some features may be more descriptive than others, e.g., some may have less class overlap and thus easier to with a linear classifier.

A BCI system that is designed to classify EEG as, say, left and right hand imagery, should preferably use some statistical features of the EEG rather than the raw (or filtered) EEG. The reason for this is the relatively small difference of the correlation between the shape of a EEG and its corresponding imagery action and that between the shape of the same EEG and any other imagery action. The stochastic nature of EEG also supports this argument since a distinct combination of inclines and

declines in the signals amplitude are very unlikely to happen at the same time for a specific imagery movement. This may, however, not be true for executed movements.

Features can also be related to other aspects such as the spatial location of the signal and the time window in which it is recorded. A description of a few of these is given below.

### 2.4.1 Spacial features

The brain has different areas devoted to different functions as was described in section 2.1.2. One may find that certain areas of the brain are more informative than others given the nature of the exercises the subject is asked to perform. Also discussed in section 2.1.2 is the location of the motor cortex which is the aft most region of the frontal lobe. One could therefore assume that this area in particular would be interesting when looking for descriptive features for *motor related potentials*. This assumption is supported by a number of texts on neuroscience [34], [8] and [38].

Researchers have tried a number of combinations of spacial features located over areas like the motor cortex, [74] and the frontal lobe, [57].

### 2.4.2 Time domain features

To this domain belong the features that are obtained by examining how a signal behaves over a period of time. A large array of signal data can be recorded during a single experiment. For example, consider a situation where six seconds of signals are recorded for each trial of an experiment at 256 samples per second in an experiment that consists of 20 trials. This is a reasonably sized set to work with but some information may be lost if one uses a measure such as the mean or variance since the samples that contain useful information may only be a small fraction of the 1536. It is therefore useful to choose a part of the signal from which information is to be extracted from. These parts are referred to as *windows* and different techniques have been applied to their selection:

- *Single window*: Involves choosing a window, the first second for example, of the trial that may contain useful information and then use the same window of the next trial.
- *Sliding window*: Slides the chosen window a predetermined amount between trials. If the window in the first trial is, for example, from 0s to 1s, then it could be from 1s to 2s in the next and so

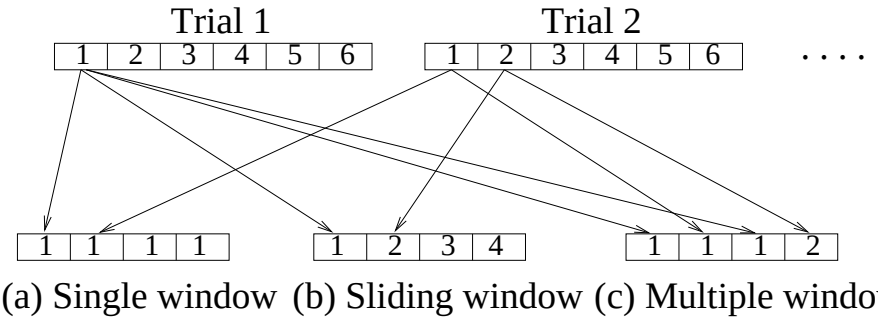



Figure 2.5: The features that can be obtained by using a (a) single window, (b) sliding window and (c) multiple windows.

on.

- *Multiple windows*: Any combination of the above can also be used to obtain more features.

The features that can be obtained using these techniques are depicted in figure 2.5.

During an experiment where visual cues are used to inform the subject of the exercise she should perform cause a response in the brain which may contain valuable information. This means that in the beginning the brain is responding to the stimuli and going into the state that it associates with the information given by the cue. The activity that proceeds voluntary action is called the *readiness potential* (RP) and is recordable and thus a potential source of information. RPs proceed conscious intentions by an average of 350 milliseconds according to [37]. This sudden burst of RP is perhaps the strongest indicator of the type of exercise that the brain will try to maintain throughout the trial and thus interesting to consider. Depending on how fast the brain detects light, such as visual cues, one could choose the first 500 milliseconds, for example, and use that as a single window of length 0.5 second in each trial. This might lead to interesting results and will be addressed later in this paper.

Once a window type and size has been determined, one can apply statistical measures on the extracted signal to find features such as the mean, variance and/or standard deviation. As mentioned earlier, obtaining these measures for a large window may reduce the amount of valuable information but they should be more descriptive for a reasonably sized window.

More sophisticated techniques have also been used for this purpose such as calculating the logarithms of the normalised variances [46] and using eigenvalues to determine the signals that maximise the difference in variance [62].

### 2.4.3 Frequency features

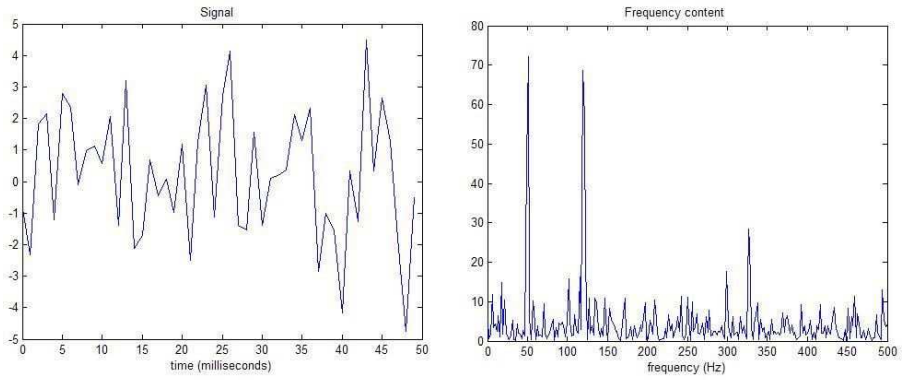
The brain is capable of producing different rhythms, some of which are dependent on the function the brain is performing, section 2.1.3. These rhythms have a frequency and amplitude which can analysed and used to obtain descriptive features. There is a certain delay between any two firings of a neuron which is called the *absolute refractory period* and lasts for up to two milliseconds after which the neuron can fire again. [2] This delay allows the neurons to synchronise their firings which they do when the brain performs certain tasks, such as motor imagery. The number of neurons that participate in this vary from task to task. This behaviour can be used as a feature by measuring the *energy* which increases when neurons fire (or don't fire) in either increasing numbers or synchronously.

A number of techniques for obtaining features from frequency components have been used in BCI systems and a few of these are described here.

#### Fourier transform

Fourier transform is a method for mapping a time domain representation of a signal into a representation in the frequency domain. [60] It's basic function is to express a function in terms of the sum of a *sinusoidal* functions multiplied by the amplitudes of the signal at different frequencies. [60] Fourier transform can be used to identify frequency components (the amplitude at different frequencies) of a signal. An example of this is depicted on figure 2.6 which shows (a) a signal in the time domain, and (b) the results of applying Fourier transformation on the signal which maps it into the frequency domain. *Parseval's theorem* can be considered a property of fourier transform and ensures that the power of the signal is equal in both domains. [21]

*Discrete fourier transform* (DFT) is fourier transform on signals where all functions are defined over discrete domains, which is inevitable when using digital computers. Further discussion will therefore only consider DFT. Formally, DFT represent a variable $x_k$ (discrete) as the sum of sinusoids:




(a) Time domain (b) Frequency domain

Figure 2.6: Two domain representations of the same signal in the (a) time domain and the (b) frequency domain, before and after fourier transform respectively.

$$x_k = \sum_{j=0}^{n-1} f_j e^{\frac{2\pi i}{n} jk} \tag{2.2}$$

for each $k = 0, \ldots, n-1$, $f_j$ are the signal amplitudes and $i$ is the *imaginary unit* ($i^2 = -1$).

Due to the complexity of the formula, applying it on large data sets is impractical, especially when considering real-time applications. A solution to this is to use the divide and conquer *fast fourier transformation* (FFT) algorithm which, according to [21], reduces the complexity from $O(n^2)$ to $O(n \log n)$. Due to the reduction in complexity that FFT offers, features can be extracted efficiently. This technique is widely used in BCI research [59] [14] [73]. The Butterworth filter described in section 2.3 can be implemented with FFT which it uses to isolate the pass frequencies. [30]

The data that is returned by the FFT algorithm needs to be converted into power per frequency. This can be done by explicitly calculating it, or by using a different approach which includes DFT in its calculations. One such is described below.

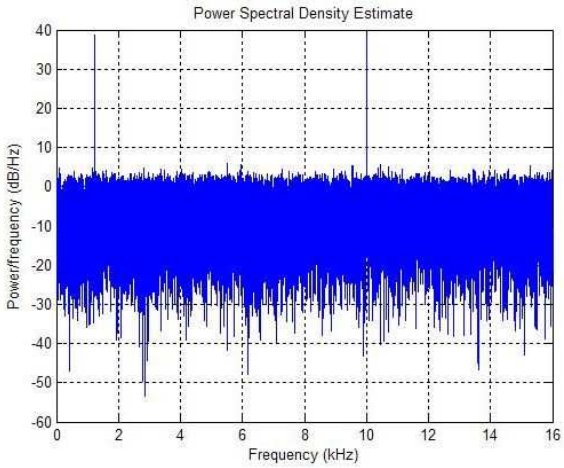



Figure 2.7: The power spectral density of a sinusoidal.

### Power spectral density

A signal, such as EEG, is composed of different frequencies with different amplitudes. The amplitude of each frequency can be measured by finding the *spectral density* of each of the frequencies which can then be used as a feature. A measure that is directly obtainable from the spectral density is the *power spectral density* (PSD) which describes the power that is carried by the signal, sometimes expressed as watts per hertz. [21]

The power spectral density of a signal can be calculated using *Welch's method* which splits the signal and calculates a *periodogram* for each of the subsets. This method is applied in, among others, [72].

An interesting relation exists between PSD and DFT. PSD calculations begin, in fact, by calculating the DFT of the signal after which the PSD is obtained using the resulting amplitudes of the DFT. One could say that PSD is simply another representation of the information that the DFT produces. This relationship is formally described in equation 2.3 where PSD of a signal is defined in the context of DFT and figure 2.7 illustrates the PSD of a sinusoidal function.

$$x_k = \frac{F_k F_k^{-1}}{2\pi} \tag{2.3}$$

where $F$ is the DFT of a signal $f$.

### Energy

The *energy* of a signal is a measure of its amplituds with respect to the zero-amplitude, i.e., amplitudes $-80$ and 80 would have the same energy. This can be calculated very easily using the equation 2.4.

$$e = \sum_{j=1}^{N} abs(x_j)^2 \tag{2.4}$$

where $x_j$ is the $j$th sample and $N$ is the total number of samples the energy is calculated for.

### Autoregressive parameters

*Autoregressive parameters* (AR parameters) are the parameters $(\phi_1, \ldots, \phi_p)$ of a *autoregressive model*[7] (AR) of order $p$. The AR model is basically a filter with an non-zero impulse response of an infinite period of time, although the AR model has more functionality such as predicting future values of a time series. [60] Equation 2.5 describes the AR model formally.

$$x_k = c + \sum_{j=1}^{p} \phi_j x_{k-j} + \epsilon_j \tag{2.5}$$

Where $\epsilon_j$ are error terms which are *independent identically-distributed* random variables [60], $c$ is a constant, $\phi_j$ are the model parameters and $p$ is the model order.

A AR model can be build using time series signals and then the parameters of that model can be applied to new signals to produce an estimate of how much the new signals diverge from the old ones, i.e., if the new could belong to the same series.

*Adaptive autoregressive parameters* (AAR parameters) differ from AR parameters by changing with time. [67] This is useful in online learning systems. Using AAR parameters as features has some advantages such as, (a) the parameters are very descriptive of the spectral properties of the signals which has the consequence that fewer parameters are required, (b) there is no need to isolate specific frequency bands, and (c) an optimum set of parameters exists that can describe the signal. [66]

[7] The *moving average* part of the model is omitted here but when included, the model is referred to as a ARMA model.

BCI research that involved extracting features using AR models include [45], [56] and [36], and AAR models [66] and [57].

Many effective techniques exist for obtaining features like the above discussion indicates and when given a set of signals, one needs to determine which features, or combinations there of, to use. Preferably they should be the most descriptive ones leading to the best classification accuracy. Consider a case where there are $d$ features, then there are $2^d$ possible feature subsets which is only a feasible number for a very small $d$. This number can, however, be reduced if one knows the exact number of features to be extracted. [6] This fact introduces a practicality limitation on performing exhaustive search in the feature space so other techniques need to be considered. Many algorithms are available which can effectively search the feature space, eliminating a subset of features at a time until the most descriptive subset remains. Divide and conquer algorithms such as *merge sort* [12] and FFT are popular for performing mathematical operations on large data sets but they are not guaranteed to try every possible feature combination. Sequential search techniques such as *sequential forward elimination* and *sequential backward elimination* are efficient ways of sequentially adding or eliminating features. [6]

Another technique is dimensionality reduction where feature vectors are mapped into a lower dimensional space through a transform function (linear in this report). One such is the *principal component analysis* (PCA) which calculates the variance of a signal by projecting it on to each axis and then returning a dimensionally reduced coordinate system with the most descriptive projection on the first axis and the second most descriptive on the second axis etc. which then correspond to the first, second, etc. *principal components*. [28] [6]

The third search technique discussed here is the *Davies-Bouldin index* (DBI) [17] which measures how much two clusters overlap. This measure indicates the extent to which the clusters are linearly separable. For a cluster $C_i$, the DBI would be calculated as follows:

$$\varphi(C_i) = \frac{1}{n}\sum_{i=1}^{n} \max_{i \neq j} \left\{ \frac{S_n(C_i) + S_n(C_j)}{S(C_i, C_j)} \right\} \tag{2.6}$$

where $n$ is the number of clusters, $S_n$ is the average distance from the elements of a cluster to its centre and $S(C_i, C_j)$ is the distance between the centres of the clusters. The limitation of the applicability of the DBI in feature selection is that it only indicates how easy it would be to separate

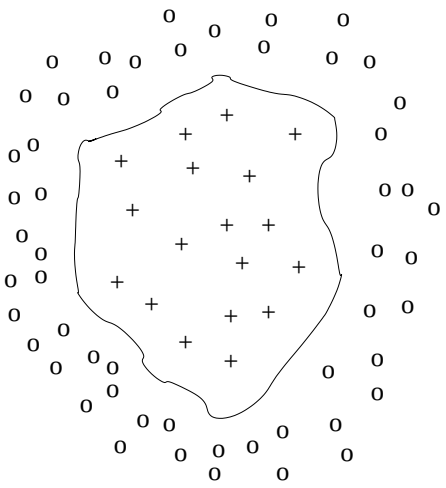


Figure 2.8: Two classes that are separable but give a high Davies-Bouldin index.

the clusters with a straight line. For example, a DBI for two clusters may be high although the clusters are separable with a polynomial curve as shown on figure 2.8. This will be discussed in more detail in section 3.3.

## 2.5 Classification of the signals

Some neuroscientists such as Hebb, McCulloch and Pitts began to research into models of the functionality of biological neurons in the early 1940s. McCulloch and Pitts [40] showed that a simplified neuron model was capable of performing elementary logical operations depending on the inputs and the threshold. Hebb [29] showed that artificial neural networks could learn by synaptic adaptation through competition, much like their biological counterpart.

An excellent description of the similarities between neural networks and artificial neural networks can be found in [2]. His arguments include, firstly, the idea of using neural network (artificial) is biologically plausible, parallel processing is a property of both and behaviour emerges through learning.

### Perceptron

A simple way of modelling the logical structure of a neural network is in the form of a *perceptron*, a term first introduced by Rosenblatt [64], which receives a number of inputs, either from other neurons or other sources and whose output is determined by the number of neurons in the network that fire. A neuron fires if the linear sum of its inputs exceeds a certain threshold. This is determined by a function $\varphi$ which can be whatever function that allows for a distinct input-output mapping. Associative memory can then be simulated by updating the threshold as well as numerical weights associated with each synapse through a learning procedure. Rosenblatt's work was a progression of earlier work done McCulloch and Pitts which was described above. Rosenblatt extended their idea to allow the model to learn although he does not include time as a factor in his theory.

A Perceptron which takes $N$ inputs is illustrated in figure 2.9 and is described formally as ...

$$v_j = \sum_{i=0}^{N} w_{ji}x_i,\ y_j = \begin{cases} 1, \text{ iff } v_j \geq \theta \\ 0, \text{ iff } v_j < \theta \end{cases} \tag{2.7}$$

where $x_i$ is the $i$th input, $w_{ji}$ is the numerical weight that is associated with the $i$th input and the $j$th neuron, and if $v_j$ is greater or equal to zero then the $j$th output, $y_j$, is 1, $-1$ otherwise. Note that $x_0$ is always 1 and $w_0$ is the bias which determines the threshold $\theta$.

When training a network using *supervised learning* [28], one has to define the desired output, from now on referred to as $\overline{z}$ and is simply the $M$ element vector $(z_1, \ldots, z_M)$, of the network when given a specific input. If the prediction of the network, $\overline{y}$, is different from $\overline{z}$, then the weights need to be updated which is done by finding the update factor, i.e., in which direction the weight vector needs to be updated to minimise the error $\overline{z}$ - $\overline{y}$. The *delta rule* [41] is one such techniques and it builds on the *Hebb rule* introduced by Hebb [29]. It elaborates on the Hebb rule in two important ways: (1) it only updates the weights when the network would have given the wrong output, and, (2) it includes the error, ($\overline{z}$ - $\overline{y}$), as a part of the rule. The amount the weights are updated using the delta rule is ...

$$\Delta\overline{w} = \alpha(\overline{z} - \overline{y})\overline{x} \tag{2.8}$$

where $\overline{(x)}$ is the input vector and $\alpha$ is the learning rate which determines the "step size" of the *gradient descent* [41].

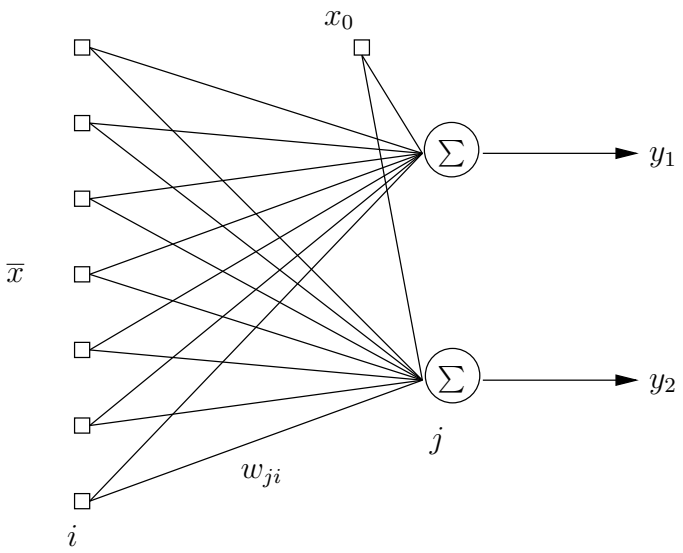



Figure 2.9: The architecture of a Perceptron that takes $N$ inputs and has two output neurons.

These properties of the Perceptron make it a linear classifier capable of separating linearly separable clusters. In fact, Rosenblatt proved this property in his *fixed-increment convergence theorem* [65].

In spite of this limitation, research [74] has shown that linear classifiers are applicable in BCI system given that the features are selected with care.

For non-linearly separable data, a more complex model is needed. A natural way of accommodating such increase in dimensionality of the data is to extend the perceptron by adding a new layer of *hidden neurons* which add dimensionality to the network. The perceptron then becomes a *multilayer perceptron* (MLP) [28] which obtains its outputs by applying a non-linear function $f$. The *hidden layer* requires a more sophisticated update procedure which is provided by the *backpropagation* algorithm [41] which builds on the delta rule. The structure and functionality of the MLP is similar to that of the perceptron so it would be more interesting to examine a technique that performs non-linear classification in a slightly different manner.

### Radial basis function network

The Radial basis function (RBF) network [28] builds on the principle that non-linearly separable data cast into high-dimensional space becomes more likely to be linearly separable. [15] A RBF network

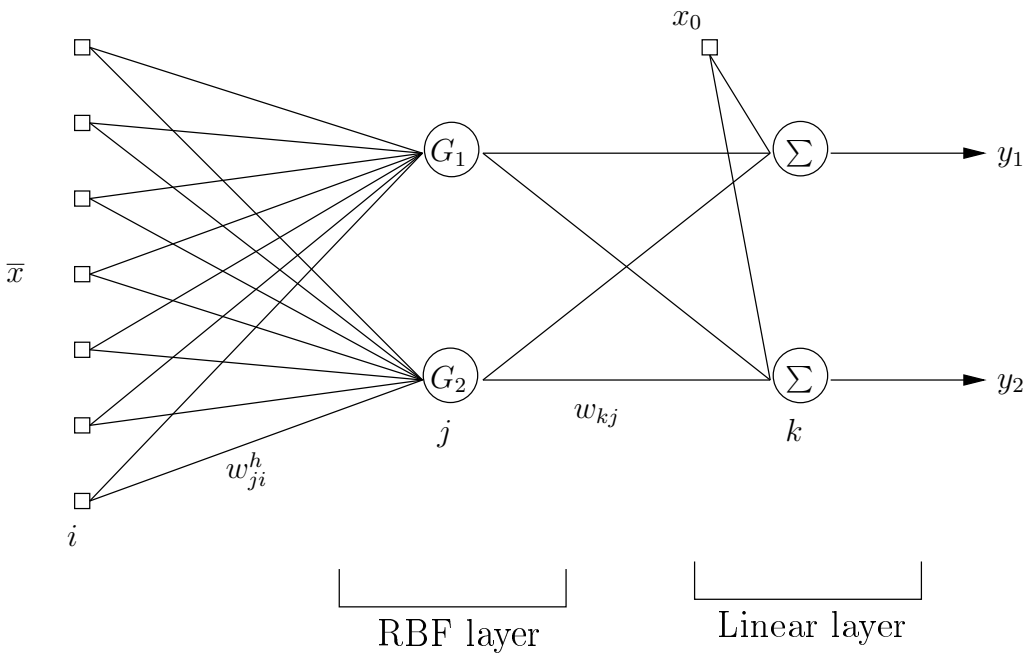



Figure 2.10: The architecture of a RBF network that takes $N$ inputs and has two output neurons.

whose structure is depicted on figure 2.10 is described formally as ...

$$y_k = \varphi \left[ \sum_j^n w_{kj} G_j(\overline{x}, w_i^h) \right] \tag{2.9}$$

where $w_i^h$ are the radial basis (RB) layer weights, $\overline{x}$ is the input vector, $G_j$ are the RB functions, also know as *hidden functions*, which can be ,for example, $G_j(x) = exp\left(-\frac{\|\overline{x}-\overline{c}\|^2}{2\sigma_j^2}\right)$. $w_{ji}$ are the linear layer weights and $\varphi$ is the *activation function* of the linear layer which, according to the threshold, determines what class should be predicted.

The RBF networks is capable of learning but the parameters that are learned are not only the weights in the linear layer but also the parameters of the RB functions such as their centres and spreads as well as their number. The whole RB layer can be trained with the backpropagation algorithm which uses $(\overline{z} - \overline{y})^2$ as an error measure, or only the weights can be trained using backpropagation and the RB function centres and spreads can be trained with, for example, a clustering algorithm such as the *k-nearest neighbour* (K-NN) [41] algorithm. The linear layer is trained using backpropagation in both

cases.

Using a clustering algorithm to train the RB function centres and spreads has several advantages. Firstly, clustering is a *unsupervised learning* [28] technique which, in theory, can find salient features in the signal that a human user might not be aware of. Secondly, it uses the input vector to train the RB layer so there is no need for expensive error calculations which may make it faster.

The output, a class prediction, from the network is a result of the classification that the linear layer performs when given the results of applying the RB functions to the network input.

### Learning vector quantisation network

The third and last classifier to be described here is the *learning vector quantisation* (LVQ) [28] network. LVQ networks have, similar to RBF networks, two layers with different functions. The first layer is the the *competitive layer* which employs the *self-organising map* (SOM) [35] algorithm, which is an unsupervised learning procedure. Figure 2.11 depicts the architecture of a LVQ network. SOM networks fall into a group called *competitive learning* networks and have the property that their neurons compete amongst themselves to respond to the input. Only one neuron will fire with any given input because they have mutually inhibitory connections, i.e., if on neuron is *on* then all others must be *off*. The neuron that wins the competition, the winning neuron, is the one whose weight makes the smallest angle with the input, i.e., the neuron for which the Euclidean distance, $\sqrt{\sum_{i=1}^{N}(w_{ji}-x_i)^2}$, from the input is the smallest. The weights of the winning neuron and its neighbours are then updated as follows ...

$$\Delta w_{ji} = \eta \cdot g_{j,j^*} \cdot (x_i - w_{ji}) \tag{2.10}$$

where $w_{ji}$ is the weight from the $i$th input to the $j$th neuron, $g_{j,j^*}$ determines the distance between each neuron $j$ and the winning neuron $j^*$ and updates the $j$th neurons weights accordingly and $\eta$ is the learning rate which decreases with time. It should be noted that the neighbourhood measure decreases as well with time thus allowing neurons that are far from the winning neuron to be undisturbed. This learning process in the SOM algorithm is called *vector quantisation* and hence the networks name.

The SOM algorithm will create $N$ clusters when using $N$ neurons which is not a problem when less than $N$ classes are required since any number of clusters can be assigned to each class. The algorithm

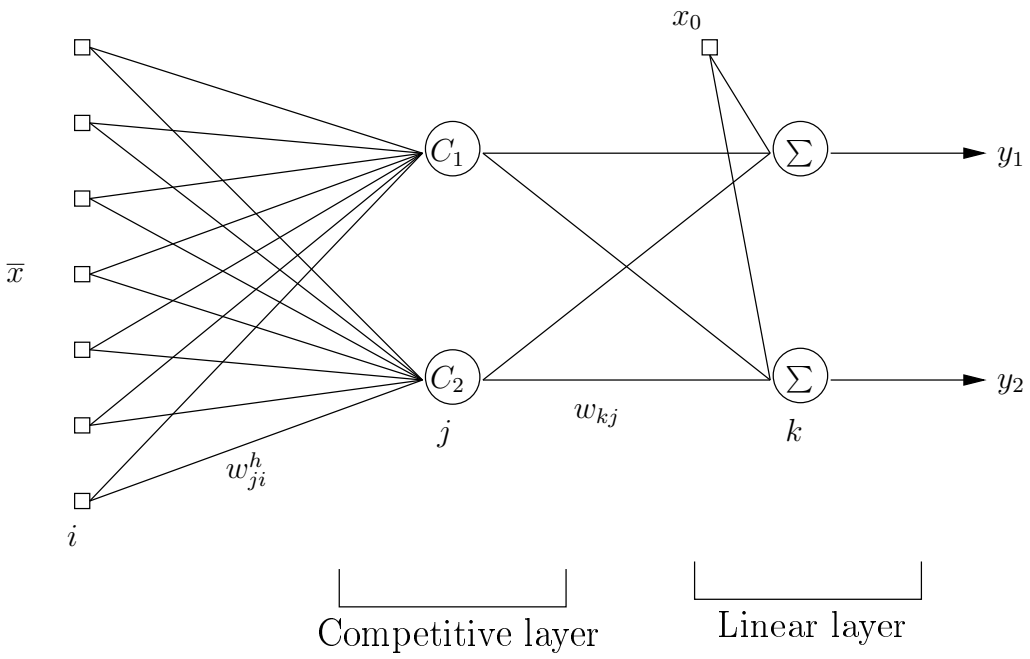



Figure 2.11: The architecture of a LVQ network that takes $N$ inputs and has two output neurons.

will, furthermore, create a two dimensional feature map where the clusters are separable.

The second layer of the LVQ network operates on the feature map that the SOM algorithm produces. It is here that the labels, the desired output $\overline{z}$, in the training data come into the picture because the linear layer uses them to improve the decision boundary produced by the first layer. The effect of this is that inputs are first clustered in a way that naturally partitions them, making them easier to separate linearly. The network will output whatever class the linear layer predicts when given the classes from the competitive layer. It is most likely this property that makes LVQ networks, and its improved relative the *distinction sensitive learning vector quantisation* network, popular in BCI research such as [54], [59] and [55].

### Training neural networks

A common problem in the application of neural networks is that they tend to learn idiosyncrasies of the training data that are not necessarily true for the whole population from which the training data is drawn from. This phenomena of *overfitting* would result in the network having better performance

on the training data than the testing data which may not posses some of the characteristics of the training data that the network has learned.

A number of techniques exists that can prevent or minimise this behaviour such as adding a *weight decay* term [41] to the learning function which keeps the weights small and makes the network slower to respond to properties that a only a small part of the data possesses.

Another more elaborate technique called *cross-validation* [28] involves dividing the training data into two sets for training and validation and the network is trained with the training set while the error is monitored by testing it on the validation set and the best performing networks weights so far are stored and returned when training is completed. The training error and the validation error will start to deviate when overfitting occurs which can be detected during learning.

# Chapter 3

# Methodology

*This chapter describes the methodology that was used in this project. It starts with describing the experimental protocol, then the signal preprocessing then feature selection and extraction and, lastly, classification. Most of the code was written in MATLAB although some was written in C++.*

## 3.1 Experiment procedure

The research described in this paper relied on very specialised equipment, except for the personal computer (PC), and thus hard to replicate if one possesses only the interest. It is therefore important to provide adequate information on this part as it may not be easily accessible otherwise. The reader should note that most of the information was obtained from the respective producers web-pages.

#### EEG-Cap and amplifier

The first step when preparing for the experiment was to make sure that EEG could be recorded without complications and too much noise. A cap with 23 electrodes was used for this purpose, 21 electrodes on the scalp, one ear reference and one ground. The two latter ones were used for later filtering which removes the noise cause by the body's natural rhythm. Figure 3.1 shows how the electrodes were positioned on the cap. The cap was a *Electro-Cap* from Allied Products [61], shown in figure 3.2 (a). Two caps were used, the E1-L for scalp circumference between 58 and 62 cm., and E1-M for 54 to 58 cm. circumference. Both caps have electrodes positioned according to the International 10-20

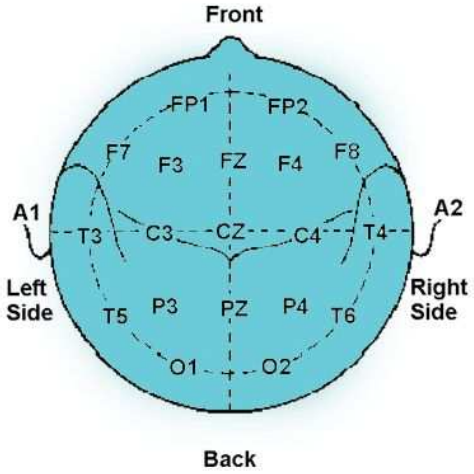



Figure 3.1: The international $10-20$ electrode setup. The letters assigned to the electrodes correspond to their relative position over the cortex and are F=frontal lobe, T=temporal lobe, C=central lobe, P=parietal lobe and O=occipital lobe. $A1$ and $A2$ are the ear references and the ground is located in front of channel $Fz$. From [68]

specification [26].

Impedance between the electrodes and the scalp was a factor when recording EEG. Using proper filtering techniques, however, could improve the quality of the EEG even though the impedance was as much as 40 k$\Omega$. [23]

The cap was then connected to an amplifier which in this case was a 24-*channel Mindset EEG Neuromapping System*, from now on simply referred to as Mindset, also provided by Allied Products. The Mindset had a 16-bit resolution, a programmable sampling rate between 64 and 1024 samples per second and active filtration implemented by two fourth order *Sallen-Key* active filters with a 50 to 60Hz stop band. The filter was provided in order to reduce interference from the power mains which in The UK is around 50Hz.

The system requirements for using the Mindset are as follows: Pentium Class or 486 PC running

Windows 95 or NT. SCSI interface (which will be discussed in greater detail later), 8 Mbs of RAM and 20 Mbs of available hard drive space.

**The SCSI** The Mindset was connected to a PC through a *small computer system interface* (SCSI) card which is a PCI device that was attached to the PC's serial bus. The Mindset could be controlled through the SCSI card but required a dedicated C/C++ library to be installed as well.

A more detailed description of the SCSI library is given in chapter 3.1.3 but a brief overview is given here. The SCSI library can be included in any C/C++ code project and provides functionality to control the Mindset to some extent. Some of the features that it implements are to start/stop sampling, error detection routines and procedures for setting various parameters of the Mindset such as the sampling rate.

Some of the code was edited by L. Citi and his modifications are used here with permission.

#### Pioneer 3-DX8 Mobile robot

The robot that was used is an ActiveMedia Robotics [63] Pioneer 3-DX mobile robot, shown in figure 3.3 and from now on simply referred to as Pioneer. It had an arsenal of utilities such as a hard disk drive, random access memory (RAM), micro-controller, lasers for navigation, bumpers, vision system, compass and wireless local area network (WLAN) as well as a gripper arm. The robot that was used in this project had a C/C++ run-time environment and a Red Hat[1] Linux operating system.

Although impressive, most of the functionality of the Pioneer went unused during the experiments which only engaged its basic functions and completely bypassed the Pioneer operating system, which will be described in more details in chapter 3.1.4.

To operate the robot one needs a connection, either WLAN or serial, and a C/C++ program compiled with the ARIA library under Linux or Windows 200.

**The ARIA** ActiveMedia Robotics Interface for Application (ARIA) is a development environment that supplies procedures that handle most of the Pioneer tasks. It had procedures for engaging/disengaging the robot, setting velocity and moving each of the two wheels forward or backward

[1] http://www.redhat.com/


(a) EEG cap (b) Mindset

Figure 3.2: The equipment that was used for the recording of EEG. (a) EEG-cap, (b) Mindset. From [61].

Figure 3.3: The Pioneer 3-DX mobile robot.

and, more primitive, reconfigure individual bits in the digital input and output which can be used to gain quick access to the Pioneer embedded computers, such as the gripper.

The primary motivation behind including the experiment in this project was to obtain familiarity with the techniques and procedures associated with EEG recording. The reason for this is that one of the goals of the project was to design a BCI system which would have a practical real-time application if implemented. This means that EEG would be recorded from a person which would try to control the mobile robot by focusing on a certain mental exercise. The experiment should thus be as similar to the real-time situation as possible which is why it was decided to use the Pioneer robot in the experiment. Apart from enhancing the credibility of the research, this approach also minimise the difference in the EEG between the experiment situation and the end situation since visually related EEG count for up to 60 percent of the EEG that is recordable from the scalp. [11]

Another motivation behind including the experiment in this project was to produce novel signals which would be subject to analysis, filtering, feature selection and classification.

Three experiments were conducted in this project with two fellow students as test subjects, subjects

S1 and S2, and one from outside the department, subject S3. There was a possibility of the subject suffering a minor injury during the setup, which is described shortly, so a voluntary consent form, appendix B.1, was borrowed from Dr. Sepulveda and modified to include the specifications of this experiment. The subjects were all healthy males of ages 20 to 25.

### 3.1.1 Experiment setup

The experiments were conducted in the Brooker laboratory of the Essex University Department of Computer Science. The laboratory is quite large considering the spatial requirements of the experiment and other students would frequently be present. They all showed considerable consideration during the experiments by constraining themselves from producing acoustic and visual disturbances. The experiments proceeded therefore without interference and with the test subjects seeing only stationary objects apart from the Pioneer robot which was moving.

**Preparing for the experiment** was a task that required concentration and experience. Dr. Sepulveda was therefor present during the setup which could last from three quarters of an hour to an hour and a half depending on the impedance between the electrodes in the cap and the test subjects scalp. A typical preparation would proceed as follows:

1. The subject was placed in a padded chair with adequate arm, leg and neck support to ensure comfort. The subject could not see the Mindset, the PC or related equipment other than the Pioneer robot. The experimenter was seated behind the subject where the experiment could be conducted through a PC.
2. Two measurements were made on the subjects scalp:
   (a) The scalp circumference which determines which cap would be best suited.
   (b) The distance from the top of the nose bone (*naison*) to the tip of the aft most region of the scalp (*inion*). This distance would be approximately 10 times the distance between the top of the nose bone to the front of the cap.
3. The cap was fitted on the subjects scalp and ears cleaned and fitted with ear-reference electrodes.
4. A strap was placed around the subjects chest which the cap was attached to. This would ensure that the cap would mostly stay in place.
5. The electrodes were *O*-shaped so that conductive gel could be injected in between each electrode and the subjects scalp. The electrodes were fitted with a *O*-shaped sponge which confined the gel to the area between the electrode and the scalp as well as providing comfort. The gel was

| Robot action | Imagery movements |
|---|---|
| Turn right | Imagine moving right arm and wrist |
| Turn left | Imagine moving left arm and wrist |
| Move straight forward | Imagine moving both legs by lifting up the heel |
| Move straight backward | Imagine opening and closing the mouth |

Table 3.1: The four directions of the Pioneer robot and their corresponding imagery movements.

injected with a flat-tipped needle which was also used to scrape dead skin of the scalp in order to reduce impedance between the electrode and the scalp. This exercise could result in a minor injury, hence the voluntary consent form, which the subjects were informed about prior to the experiment.

6. Impedance was tested using the ground channel as a reference and the aim was to get it no higher than 7 kΩ so the needle was used to improve conductivity when required.

7. Once the ear-reference and all 21 electrodes were tested for impedances then the setup was complete as far as regards the subject.

8. The Pioneer robot was placed in front of the chair with its back facing towards the subject. The distance between the subjects eyes and the robot was 150 cm. in all experiments.

The experiment protocol was explained to the subject during and after the setup. This was a difficult task because there are many things that the subject needed to consider with the least possible effort. The protocol is described below but first a few words on the two mental exercises associated with the experiment.

**Mental exercise** 1 included imagery movements of four distinct parts of the body that were associated with each of the four actions of the Pioneer robot and are described in table 3.1. The reason for choosing these body parts in particular was that they are all motor related and thus controlled by the brains motor cortex, see section 2.1.2. Another benefit is that the parts of the motor cortex, the primary motor area to be exact, that control these body parts can be located on the scalp, figure 2.3, as well as having large areas of the primary motor area associated with them. A similar approach was taken in [57].

**Mental exercise** 2 required the subject to visualise the potential being carried from the motor cortex to the muscles in the body part that correspond to the robot's direction which are the same

as in exercise 1. This approach will be referred to as *signal imagery* in the remainder of this paper and is a first. This is different than motor imagery in the sense that the average person may not have any knowledge of the mechanisms that are at work when the brain transmits signals to the rest of the body. It might be that the signal imagery gives better readings because of how close the brain comes to actually performing the movements compared to motor imagery. It was decided to use two different exercises and then compare their relative success. Finding a candidate other than motor imagery proved difficult when dismissing all exercises that involve executed movements. The reason for excluding these is that the techniques adopted here should also work for people which suffer from total paralysis and do not have any voluntary control over their muscles. Only one alternative [50] to motor imagery was found and that involved arithmetic operations which were not considered a very intuitive way of controlling a mobile robot. Exercise 2 was therefore formulated and adopted in spite of its lack of scientific credentials to provide an alternative to a well established technique.

Exercises that involve visualisation and imagination of activities are not guaranteed to be interpreted in the same way between subjects. It was therefore made very clear that when, say, a right hand imagery was required, then it would not be sufficient to simply repeat the words "move right arm" to oneself but instead try to come as close to executing the action as possible without engaging any muscles.

### 3.1.2 Experiment protocol

It was important to consider how to communicate to the subject what the next action of the robot would be during the experiment. The reason for this is that it allowed the subject to begin a planning process which may have produced informative EEG that could later approve classification accuracy. It was of great importance how these cues are produced as the discussion in section 2.2 indicates. It was decided to use visual cues instead of acoustic ones which was based on the fact that acoustic information takes longer to be processed by the brain than visual information. Four light-emitting diodes (LEDs) that were positioned on the top of the robot were used as cues. The board was constructed by Mr. Dowling which is a member of the technical staff in the University of Essex. The LEDs were positioned to the right and left, front and back, corresponding to the four directions that the robot could move

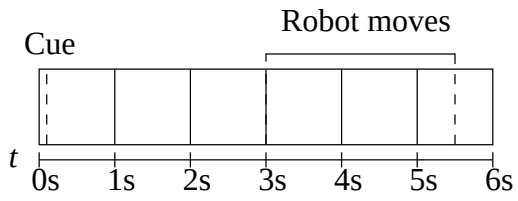



Figure 3.4: The time line for each trial of the experiment.

in. The cue would be that one of the LEDs would blink.

The protocol that was used in the experiment had two main parts which were the setup and the actual experiment where timing and other factors were crucial. The latter part will now be described in more detail as it was explained to the subjects.

1. All four LEDs blink twice indicating the beginning of the experiment.
2. EEG recording starts.
3. The following is performed 20 times (20 trials).
   (a) Relax for 6s.
   (b) A LED blinks indicating the start of the trial and the next action. Subject performs exercise.
   (c) Continues the exercise for 3s.
   (d) Pioneer robot moves.
   (e) Continues the exercise and follows the robot for 3s.
   (f) All four LEDs blink to signal that the subject can now relax and prepare for the next trial.
4. EEG recording stops.

Each iteration of the loop in the protocol constituted one trial. There were a total of 20 trials in one sequence which there were two of, both containing five trials for each action but in different orders. These two alternated so that the subjects did not become preoccupied with trying to predict the next action between trials. This was accomplished by producing a random sequence of 40 trials containing 10 trials for each action and then dividing it while ensuring that a total of five trials per action remaind in each part. Furthermore, each sequence ran four times for each exercises. This is summarised in table 3.2 and figure 3.4 shows the time line.

Table 3.2 shows the time requirements for each sequence of the experiment which adds up to 1920 seconds in total, or 32 minutes. This number is obtained by omitting the delays that were between

| Exercise | Sequence # | Trials | # of Trials | Seconds |
|---|---|---|---|---|
| Movement imagery | 1 | 1 - 20 | 20 | 240 |
| | 2 | 21 - 40 | 20 | 240 |
| | 1 | 1 - 20 | 20 | 240 |
| | 2 | 21 - 40 | 20 | 240 |
| *Intermediate* | | | 80 | 960 |
| Signal imagery | 1 | 1 - 20 | 20 | 240 |
| | 2 | 21 - 40 | 20 | 240 |
| | 1 | 1 - 20 | 20 | 240 |
| | 2 | 21 - 40 | 20 | 240 |
| *Total* | | | 160 | 1920 |

Table 3.2: The division of trials between sequences as well as the duration of each sequence and the whole experiment (omitting breaks).

sequences and any breaks that the subject requested. Adding this to the time required for the setup, 45 - 90 minutes, and the approximated total delay (and breaks) of 30 minutes between sequences, the duration of the whole experiment procedure was expected to be between 107 and 152 minutes which proved reasonably accurate.

### 3.1.3 EEG recording

Section 3.1 described the SCSI board and the SCSI library which needs to be included in applications which make use of the Mindset functionality. The SCSI library was obtained from the producer and later modified by L. Citi to enhance its generality.

Earlier experiments by Dr. Sepulveda required an application which could display four commands on a screen, one at a time, in a predetermined sequence. The code for this application was used in this project and modified with permission. The code (appendix C.1) was modified to implement an algorithm which would conduct each sequence of the experiment. It was modified to follow the logical structure of the experiment protocol with a few exceptions and additions. An outline of the algorithm is given below as it was written in C++.

```
blinkAll();
blinkAll();
Acq.SCSI_StartSampling();
int i=0;
int count=0;
```

```
bn = 0;
Sleep(6000);
for(i=0; i<20; i++) {
    Cue();
    ... The below is done three times ...
    samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
    bn++;
    Sleep(1000);
    ..........................................
    samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
    bn++;
    robot -> setVel(leftvel, rightvel);
    cprintf("%6s",seq[count]);
    ... The below is done three times ...
    samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
    bn++;
    Sleep(1000);
    ..........................................
    samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
    bn++;
    BlinkAll();
    count++;
    Sleep(6000);
}
```

The procedures `blinkAll()` and `setVel(leftvel,rightvel)` turned all four LEDs on for one second and set the Pioneer robots velocity, respectively. They will be described in more detail in section 3.1.4. `Acq` was an instance of the SCSI class which called the SCSI libraries and provided an interface to the Mindset. The `Cue()` procedure turned one LED on for one second corresponding to the direction the robot would move. The direction was also printed on the screen for the experimenter to follow. The `Sleep(x)` procedures maked the process sleep for $x$ milliseconds. The most interesting addition was the data that were stored in the `samplmark` array. The data were numbers that indicated which sample was being recorded from the Mindset when the `SCSI_GetCurrentSampleNum()` procedure was called. This happened eight times in each trial and is described in more detail in section 3.2. These markers were used in the preprocessing to extract only meaningful data from the whole sequence which, in practice, also recorded EEG while the subject relaxed. These markers were also useful when the EEG was extracted from, say, the first second of each trial. This will be described in more detail in section 3.2.

| Procedure name | Arguments | Description |
|---|---|---|
| **run** | (true \| false) | Initialises the robot. |
| `setVel2` | (left wheel velocity, right wheel velocity) | Sets the velocity of each wheel individually as specified by the arguments. |
| `com2Bytes` | (Command name, lower bit, higher bit) | A low-level procedure which was used to implement high-level functionality such as setting the velocity. It allowed the user to set individual bits in the digital input/output ports according to the lower and higher bit arguments. |
| `blinkAll` | () | Sets the bits in the digital input/output ports that tured the LEDs on and off. |

Table 3.3: A description of relevant ARIA procedures.

### 3.1.4 Robot control

The ARIA library, section 3.1, provided procedures that controlled the Pioneer robot. An instance of the ArRobot class had to be created which would make all of the robot's functionality available. Once the robot was initialised it could be controlled directly by calling the ARIA procedures which are described below.

One of the Pioneer robot's accessory was a gripper arm which was disconnect and the ports that would otherwise control the gripper arm were assigned to the LED board. There were some problems encountered while determining the bit masks specifications so the ports that were used in the application were found through trial and error.

The robot was connect to the PC through a serial connection which had the advantage of engaging only the robot microprocessor. The robot could then respond to commands as quickly as possible. An alternative would have been to use a wireless network connection which would have been somewhat slower and since the timing is a crucial factor in an experiment of this kind, a serial connection was used.

## 3.2 Signal preprocessing

Large amount of signals were recorded during the experiment like the previous section indicates. The signals were stored in binary files which were converted to an array representation with a script (appendix C.2) that was created at Essex University. Those files were structured as a 24 row array which corresponded to the 24 electrodes that can be connected to the Mindset although only 21 of them were used which is the number of electrodes on the cap.

There were a few issues that needed to be addressed in order to structure the signals in a way that would make it more intuitive to use in the feature extraction algorithms. These are ...

- Each experiment produced 16 binary files that contained the EEG recordings. They were named according to the type of exercise and sequence number and contained equal numbers of trials for each of the four actions. This requires a method of grouping signals together that belonged to the same action.
- The files also contained signals that were recorded during relaxation and would not be used further, so these segments needed to be discarded.
- A separate file contained the markers which had to be included in the data for feature extraction, e.g., when extracting a one second segment out of the six second trial data.

A simple algorithm was implemented in MATLAB [39] (appendix C.3) which extracted the signals from the binary files and stored it in arrays until all the signals that corresponded to, say, a right hand movement imagery over four sequences was gathered. After which, the array was stored in a file that contained only signals for one type of action. This was done separately for each exercise.

The markers were stored in separate files and used by the algorithm in parallel with the signal arrays. To eliminate any signals recorded while the subject relaxed the algorithm simply counted the number of markers and deleted all signals after it counted eight markers and until the next marker was encountered, then it counted another eight etc. The algorithm added one row to the new signal arrays which then had a total of 25 rows and each time the algorithm encountered a column that corresponded to a marker it added a 1 in this additional row. When the algorithm completed it had put a 1 in the 25th row whenever the column corresponded to a marker, and a 0 otherwise. Upon completion, the

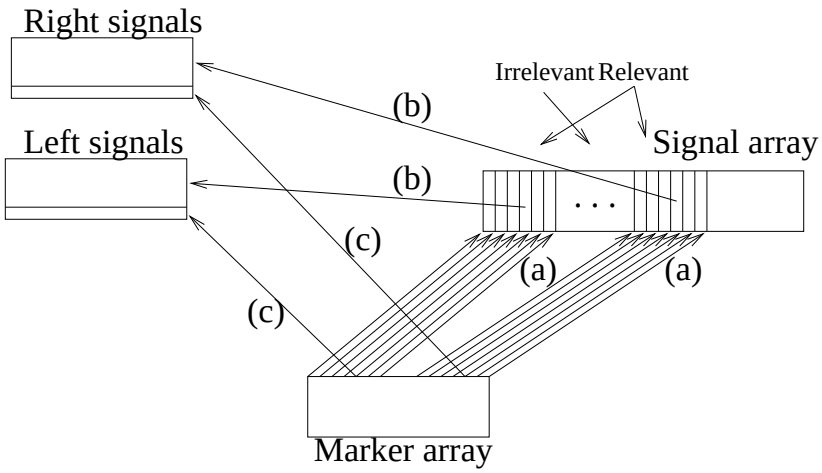



Figure 3.5: The figure shows how signals were sorted after the experiment. The process includes (a) locating relevant signals using markers, (b) grouping relevant signals together, and (c) adding the markers to the array.

algorithm had successfully partitioned the signals into files containing EEG recordings for each action and each exercise as well as including the markers directly in the new arrays, in row 25.

The operations of the algorithm are depicted in figure 3.5.

### 3.2.1 Filtering

The Butterworth filter was used to bandpass filter the signals, section 2.3. The actual passband was considered a feature and will thus be described in section 3.3. The reason for this choice was that the Butterworth filter has a minimal loss of information because of a smoother frequency response in the passband than, say, a Chebyshev filter, see figure 2.4. This property of the Butterworth filter made it a very attractive choice since EEG in general contain much noise to begin with so reducing it further requires a reason that builds on extensive experimentation where the lost information is shown to be non-descriptive.

The filter was created with MATLAB's `butter` function which belongs to the *signal processing toolbox*. Two arguments were supplied to the function which were the filter order and low/high-pass

frequencies. The filter order $n$ argument should only be a fourth of the desired filter order for the following reasons. Firstly, because the filter performed bandpass filtering it would, in effect, make two passes over the signal, one for highpass and one for lowpass resulting in filtering of order $2n$. This had the same effect as applying two filters, low and high pass, of order $n$ each. Secondly, the MATLAB function `filtfilt` was used to apply the filter on the signal. This made the filter pass over the signal twice, once in each direction. The reason for this choice was that the first pass, which could be invoked with the function `filter`, shifted the signal slightly in the direction of the filtering which might have resulted in some information loss. The `filtfilt` function minimised this effect by making another pass over the signal in the reverse direction, thus shifting the signal back to its original position. The resulting filter would thus be a *zero-phase* [30] filter of order $4n$.

The second argument that was supplied to the `butter` function was the normalised range of the passband. The sampling frequency was 256 so the passband range was normalised between 0 and 128, where 128 is the *Nyquist frequency*, i.e., half of the sampling frequency. [60].

## 3.3 Feature selection

The importance of using a feature selection criteria was briefly discussed in section 2.4 (page 23). The main reason for this is that the feature space was huge and exhaustive search would be very impractical in most situations. The criteria that was used here is the Davies-Bouldin index (DBI) which measured class overlap and hence how easy it would be to classify the signals with a linear classifier. The signals, however, were expected to be non-linear which would give a high DBI even when easily separable by a non-linear classifier. A consequence of this was that the signals were subject to non-linear transform using a radial basis function, MATLAB's `radbas`, after which the DBI was calculated. The signals, when cast into higher-dimensional space, would then be more likely to have a lower DBI than before the transform, perhaps even low enough to be linearly separable. [15] Furthermore, the DBI was readily calculated and thus a good choice due to project timing constraints.

The DBI and the post transform DBI were calculated for the features which are summarised in

| | |
|---|---|
| **Exercise** | Movement imagery, signal imagery |
| **Filter orders** $(4 * n)$ | n = {1, 2, 3, 4, 5} |
| **Spatial: channels** | 1,. . . ,21 |
| **Time: single windows (seconds)** | Whole trial $(6s)$, $0-0.5$, $0-1$, $1-2$, $2-3$, $3-4$, $4-5$, $5-6$ |
| **Time: raw signal** | Mean, variance, std., maximum and minimum |
| **Frequency: bands (Hz)** | $0.5-45$, $0.5-3.5(\delta)$, $3.5-8(\theta)$, $8-13(\alpha)$, $13-22(\beta)$ and $22-40(\gamma)$ |
| **Frequency: power spectral density** | Mean, variance, std., maximum and minimum |

Table 3.4: This table gives a summary of the features that the DBI was calculated for. Those features that were associated with the lowest DBI for each category would then be selected and used for classification.

table 3.4.

Features were eliminated sequentially so an initial configuration was established which was used in all but the last DBI calculations or until the features that had the lowest DBI were found. This configuration was a follows:

- All 21 channels.
- Time window $0-6$, a whole trial.
- Raw signal mean.
- Filter order 3, 12 after four passes.

At first the exercises were subject to evaluation using the initial configuration for both movement imagery and signal imagery. The one that gave the lowest DBI on any channel would then be used in further evaluations. This process continued until the features that had the lowest DBI were found. It was not believed to be problematic to conduct the evaluations in this way because all three networks had neurons whose number surpassed the number of features. A neuron effectively adds a new dimension to the space where a decision hyperplane would be drawn. A property of DBI is that adding many zero-DBI features, i.e., they form clusters which are separable by a straight line, together would result in a collection of clusters that could be separated by a straight line in each dimension.

### 3.3.1 Feature extraction

Once the features were selected, they needed to be extracted - sometimes calculated - in an effective way. Some of these methods were trivial such as extracting signals from a specific channel which was simply a row vector in the signal array and the signals for a specific exercise were simply loaded into the array that was used throughout the program. The filter order was simply increased with a counter variable. A function was created that called the MATLAB functions that implemented the filter ...

- `bandpass(Input, filter order, lowpass frequency, highpass frequency, channel range)`, see appendix C.4.

To extract a specific frequency band, the $\alpha$ band for example, one only needed to supply 8 and 13 as low- and highpass frequencies.

The mean, variance, standard deviation, maximum and minimum were also trivial to calculate in MATLAB with their respective functions `mean`, `var`, `std`, `max` and `min`.

Extracting a signal contained in a specific time window required that an algorithm was devised which could used the markers in the 25th row of the array to determine which segment to extract. This algorithm was implemented in the function `extractsegment` ...

- `extractsegment(Input, start column, window number)`, see appendix C.5.

Window number 1 would be between the first and second markers and contain one second of EEG recordings.

Calculating the PSD of the signal was made very easy by MATLABs `pwelch` function which accepted a row vector as argument and returned the power spectral density of its input. The average power of the signal was calculated with `dspdata.psd`, another MATLAB function, which accepted a vector containing the PSD estimation data. This made it all very intuitive to use and a function was created that did this automatically when given a row vector ...

- `psd(Input)`, see appendix C.6.

The energy was calculated using equation 2.4, page 22, when given a row vector ...

- `calculateenergy(Input)`, see appendix C.7.

## 3.4 Pattern classification

The classifiers described in section 2.5 were all used in this project and their implementation and specifications are described below, but firstly, the reasons for choosing these classifiers are as follows ...

1. The perceptron is a linear classifier which has been shown to produce good results when used in a BCI system, e.g., [4] and [74].
2. A linear classifier such as the perceptron would be a very valuable commodity in a real-time implementation of this system if used for a wheelchair controller where the time it would take the system to response to the users intentions is directly related to her safety.
3. The RBF network does not share the linear classifiers popularity in BCI research but it provided an insight into how applicable it would be to only perform non-linear transformation of the signals which is a relatively simple calculation. The same argument of user safety applies here.
4. The LVQ network is perhaps the most widely adopted classifier in BCI research, e.g., [55], [54] and [59].
5. The LVQ network performs clustering on the signals and thus well suited to find salient features in the signals that a human inspector might not notice.

The performance measures common to them all was the average classification accuracy and the average mean square error (MSE).

### 3.4.1 Perceptron

A perceptron was created in MATLAB with the function `newff`[2]. This function accepted parameters that defined some of the structural and learning parameters. These were, firstly, the minimum and maximum values of the input which was calculated using MATLABs `minmax` function. Secondly, the number of neurons, which is described below. The third and fourth arguments were the activation function and the learning function, respectively, both of which are described below. Other related aspects were considered as well and a brief description of each of them is given as well.

[2]Alternatively, `newp` could have be used but fewer network parameters could then be specified.

**The Number of neurons** was determined manually by comparing the performance of networks with different numbers of neurons. The optimal number of neurons, found to be 15 in this case, depended on the complexity of the function to be approximated. To many neurons may have led to overfitting, see section 2.5, and to few may have caused underfitting - insufficient learning.

**The activation function** that was used is a linear transfer function calculated with the MATLAB function `purelin`. More sophisticated activation functions were available but these are used in MLP networks and not applicable in a linear separation unit such as the perceptron.

**The learning function** that was used is `learngd` which is a gradient descent weight and bias learning function. A number of learning functions were available in MATLAB and alternatives include `traingd` and `traingdm` which are both gradient descent with backpropagation learning functions with and without a *momentum* [41] term. These are, however, for use with multilayer networks.

**The learning rate** was, unfortunately, not subject to extensive testing but set as MATLABs default value of 0.05.

**Early stopping** was used during training to prevent overfitting, see section 2.5. Overfitting prevention procedures are not implemented in MATLAB by default so this had to be done explicitly by splitting the training set into two subsets and instruct MATLAB to perform cross-validation, section 2.5, page 29.

**The training epochs** were set to be 300 although cross-validation would sometimes suspend training after fewer epochs.

**Regularisation** is another technique for preventing overfitting and involved changing the performance function used by the network which would be the sum of squares error that the network produces on the training data set. The performance function was modified to include the mean of the sum of squares of the network weights and bias which resulted in smaller weights and bias and thus

forcing the network response to be smoother. [30] This was done manually in MATLAB by setting `performFcn` to be `msereg`. This can be seen in the code example below.

**Code example**

```
net = newff(minmax(Train),[15],{'purelin'},'traingd'};
net.trainParam.lr = 0.05;
net.trainParam.epochs = 00;
net.performFcn = 'msereg';
net.trainParam.goal = 0.001;
net = init(net);
net = train(net, Train, Target, [], [], Val);
Y = sim(net, Test);
error = mse(TestTarget - Y);
```

**Training** consisted of three individual runs where a new network was created and tested, each with a different portion of the signals. Then the average accuracy and error was calculated.

The training, validation and testing signals were stored in the arrays `Train`, `Val.P` and `Test` respectively, containing a third of the whole collection each. The `Train` array was associated with a target class array `Target` which contained the desired output. The `Test` array was also associated with a target class array `TestTarget` which was used to measure the performance of the network. MATLAB required that the validation signals be stored in a structure, hence the `Val.P` where `Val` is the structure and `P` is the actual array. Furthermore, it was also required that the structure contained a target class array which was `Val.T`.

The size of the arrays depended on the number of features, $F_1, F_2, ..., F_N$, and was implemented as the transpose of the row vector containing these, i.e., $[F_1, F_2, ..., F_N]^T$. The targets were designed as column vectors containing one number for each class and set to 1 if the corresponding output neuron should learn to predict that class, $-1$ otherwise. Consider two output neurons learning to predict a class each, then the target could be $[1, -1]^T$ for one class and $[-1, 1]^T$ for the other.

The performance was measured by calculating the mean square error between the predicted output and desired output. This was done with MATLAB's `mse` which argument is an array containing the desired output subtracted from the predicted output, $\overline{z} - \overline{y}$.

### 3.4.2 Radial basis function network

A RBF network was created in MATLAB with the `newrb` function. The arguments supplied to this function were the training and target arrays, an error goal and the spread of the RB functions. The training procedure of the MATLAB RBF network was different from that of, say, a perceptron, as the RBF network is not trained explicitly. This is described below in more detail.

**The activation functions** in the RBF network are `radbas` in the RB layer and `purelin` in the linear layer. The `radbas` function takes as input an array of column vectors and outputs each element after it has been passed through a radial basis function which casts the input into higher-dimensional space. The `purelin` function is the same as is used in the perceptron.

**Code example**

```
net = newrb(Train, Target);
Y = sim(net, Test);
error = mse(TestTarget - Y);
```

**Training** happened right after the network creation without requiring any explicit initialisation by the user. The network started with only two neurons, one in the radial basis layer (RB neuron) and one in the linear layer. The network was then trained by updating the following parameters:

- RB neurons were added to the RB layer through an incremental process where each new neuron was created with weights equal to the input vector that had the greatest error, i.e., the input vector that the network had the most difficulty classifying.
- RB function centres and spreads could have been learned using either backpropagation or clustering as was described in section 2.5, page 26. MATLAB used the backpropagation algorithm by default which was left unchanged.
- Network weights in both layers were learned using backpropagation.

The training would proceed in a somewhat different manner from the one used for the perceptron because MATLAB does not give the user many options when creating a RBF network which, unfortunately, led to a situation where early stopping could not be implemented for the RBF network. To

compensate for this, and to ensure the validity of later comparisons of the networks, another recognised technique was used when training the RBF network.

The technique involved dividing the signals into two subsets which were used for training and testing, containing 4/5th parts and 1/5th part of the signals respectively. This was done 5 times in total and the testing signals were drawn from a different 1/5th part each time. This approach resembles 10-*fold cross-validation*, cross-validation performed 10 times with a different partitioning each time, but not quite as elegant which is due to timing constraints and only done 5 times because of the limited number of signals.

The training and target arrays for this network had the same structure as the ones used to train the perceptron except there was no need for the `Val` structure.

### 3.4.3 Learning vector quantisation network

A LVQ network was created in MATLAB in a similar way as the perceptron, i.e., more parameters could be specified prior to training. The network was created with the function `newlvq` which takes as arguments an array of minimum and maximum values for each row in the input, a number of neurons in the competitive layer and a vector of the typical class percentage. The learning rate and learning function could also be specified and they will be described shortly.

**The neurons** in the competitive layer need to be at least as many as the target classes. [28] The reason for this is that the predictions made by the linear layer depend on which neurons in the competitive layer fire. Having more neurons than classes in the competitive layer increased the generality of the network and thus reduced the risk of the network overfitting the data. The linear layer had a neuron for each target class, each of which was associated with all of the competitive layer neurons.

**The activation functions** that were used to determine the output from each layer were `compet` in the competitive layer and `purelin` in the linear layer. The `compet` function accepted one input which

is an array of column vectors and returned a vector for each of those with 1 where the vector had its maximum value and 0 in the other places. An example of this is the input vector $[1, 2, 4, 3]$ which would result in the output vector $[0, 0, 1, 0]$ if passed through the `compet` function. The `purelin` is the same as in the two other networks.

**The learning function** was, by default in MATLAB, `learnlv1` which propagated the error back to the competitive layer once the predicted class had been compared to the desired target class. In a LVQ network, the competitive layer weights were considered to be arrays rather than numbers and the learning involved shifting the content of the array to match the output of the linear layer. Consider an example of the learning function operation where neuron $i$ in the competitive layer was the only one that fired, then the $i$th row of the competitive layer weight array would have a one in it, all others would have zero. This row would then be shifted towards the input vector if the prediction turned out to be correct, shifted away from it otherwise. This implemented a pretty straightforward concept of grouping competitive neurons together that predicted similar classes.

An alternative to `learnlv1` was the `learnlv2` learning function which also considered runners up in the competitions and allowed the user to specify weight gradients and neuron distances. This functionality was not required to implement a working LVQ network so the network was implemented with `learnlv1`.

**Class percentage** was simply a vector describing the percentages of training examples that belonged to each class. In this case where two classes were used, the vector was $[0.5, 0.5]$.

**The learning rate** was not subject to much experimentation. The default value was 0.01 and was not changed.

**The training epochs** were 100 which was established by trying different values and comparing the performance of the networks.

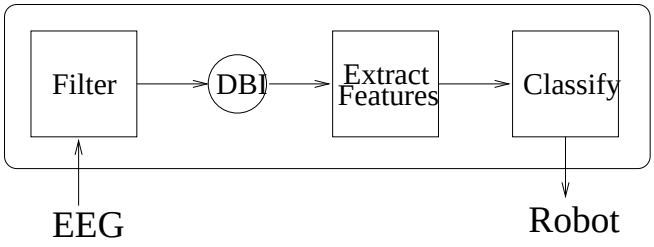



Figure 3.6: An outline of the design.

**Code example**

```
net = newlvq(minmax(Train), 18, [0.5 0.5]);
net = trainParam.epochs = 300;
net = trainParam.goal = 0.001;
vTarget = ind2vec(Target);
net = train(net, Train, Target);
vY = sim(net, Test);
Y = vec2ind(vY);
error = mse(TestTarget - Y);
```

**Training** of the LVQ network was similar to the one in the RBF network, i.e., a primitive implementation of the 10-fold cross-validation technique.

The training signals had the same structure as in the other networks but the targets were changed to contain a row vector of indices where each distinct index encoded a class. The functions `ind2vec` and `vec2ind` are a convenient way of representing indices as vectors and vectors as indices, respectively.

## 3.5 Towards the design

Most of the code presented here was written in MATLAB while the application that controls the Pioneer robot was written in C++. These would have to be combined if implemented as a real-time system which would have been readily done by using MATLAB's `mcc` compiler which could compile all of the code written for this project into either C++ source code or a *dynamic link library* (dll), both of which could be included in the C++ project and thus making the MATLAB functions available in the same application as the Pioneer robot controller.

An outline of the design is given in figure 3.6 where each of the parts of the BCI system are shown. The specifications of each part was the above description of the techniques that were used and the features the were selected.

# Chapter 4

# Results

*This chapter describes the results of the applied techniques described in chapter 3. The results are accompanied by notes on their interpretation.*

## 4.1 Experiment procedure

The experiments ran without any major complications. EEG were recorded and markers were positioned in the right places.

The subjects reported that they started loosing concentration after about an hour and a half which may have affected the quality of the EEG. If it were not for timing constraints, a reasonable solution might have been to divide the exercise into two parts or have longer, predetermined breaks. The test subjects also exhibited willingness to continue in spite of their lack of concentration in order to be finished earlier. In spite of their motivation and greatly appreciated effort, one can not expect people to have total control over their concentration, especially when tired and sitting in a comfortable chair in a quiet room for a period of time as long as was the case. This is the primary indicator that more and shorter experiments would have been in order.

The result of the experiments was a collection of EEG recordings for subject S1, S2 and S3. All these were different as can be seen in figure 4.1 but note that visual inspection is not reliable since it gives little indication of the statistical relationships between them. The signals from subject S3 were

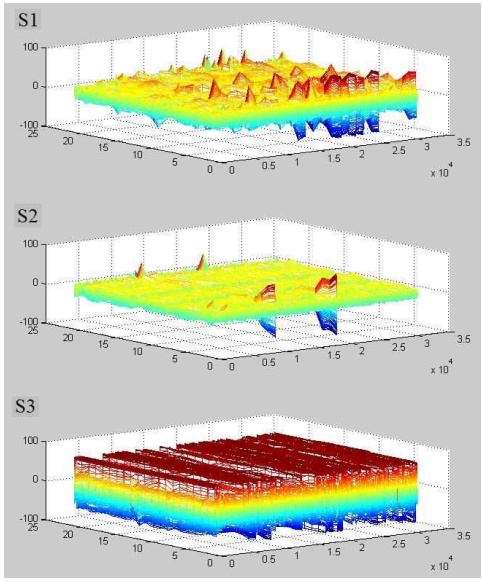



Figure 4.1: Mesh diagram of the recorded EEG for all subjects.

considerably noisy which may have been caused by muscle activity, hence the high amplitudes. Spikes of high amplitudes are otherwise mostly related to eye blinking and neck movements.

## 4.2 Signal preprocessing

The most time consuming parts of the application where the partitioning and filtering. This was not surprising considering the amount of work that was performed by these parts. The partitioning algorithm included a number of `for` loops, see appendix C.3, so the performance bottleneck was the PC's processor. All signals were stored in the PC's primary memory so the algorithm operations could acquire them quickly. The same was true for the filter whose speed depended mostly on the PC's processor. This was not expected to be a problem in a real-time situation since the number of signals would be considerably less and thus more rapidly preprocessed.

| Domain | Feature | DBI | DBI(RB) |
|---|---|---|---|
| Exercise | Movement | 4.04 | 7.99 |
| | Signal | 3.60 | 4.00 |
| Spatial | Channel $Fp1$ | 3.60 | 12.43 |
| | Channel $Fp2$ | 4.70 | 16.65 |
| | Channel $F7$ | 11.75 | 4.69 |
| | Channel $F3$ | 4.96 | 4.78 |
| | Channel $Fz$ | 4.68 | 12.03 |
| | Channel $F4$ | 4.37 | 7.06 |
| | Channel $F8$ | 8.00 | 52.96 |
| | Channel $T3$ | 9.97 | 527.62 |
| | Channel $C3$ | 7.38 | 5.45 |
| | Channel $Cz$ | 8.53 | 4.355 |
| | Channel $C4$ | 12.89 | 7.75 |
| | Channel $T4$ | 9.38 | 83.05 |
| | Channel $T5$ | 41.28 | 6.04 |
| | Channel $P3$ | 38.36 | 4.00 |
| | Channel $Pz$ | 42.03 | 4.59 |
| | Channel $P4$ | 64.67 | 6.75 |
| | Channel $T6$ | 25.85 | 11.52 |
| | Channel $O1$ | 47.92 | 5.70 |
| | Channel $O2$ | 225.81 | 5.71 |
| | Channel $Fpz$ | 4.07 | 8.41 |
| | Channel $Oz$ | 107.14 | 4.39 |

Table 4.1: The DBI on which feature selection was based. (Continued in table 4.2)

## 4.3 Feature selection

The DBI were calculated for the features that were selected in section 3.3 and are summarised in tables 4.1 and 4.2. Accompanying the DBI for the features is the DBI which were calculated after the signals were transformed with a radial basis function, this is column `DBI(RB)` in the tables.

A summary of the features that were selected is given below. These features were used to train and test the classifiers.

- Signal imagery.
- Channels $Fp1$, $Fpz$, $Fp2$, $F3$, $Fz$ and $F4$.
- Window $0 - 1$.

| Domain | Feature | DBI | DBI(RB) |
|---|---|---|---|
| Time | Window $0-6$ | 3.60 | 7.06 |
| | Window $0-0.5$ | 4.10 | 8.54 |
| | Window $0-1$ | 3.35 | 9.85 |
| | Window $1-2$ | 14.82 | 12.75 |
| | Window $2-3$ | 14.14 | 9.61 |
| | Window $3-4$ | 15.79 | 6.03 |
| | Window $4-5$ | 8.17 | 13.22 |
| | Window $5-6$ | 5.53 | 7.50 |
| | Raw signal mean | 3.35 | 9.85 |
| | Raw signal var. | 8.62 | 6.24 |
| | Raw signal std. | 11.24 | 4.46 |
| | Raw signal max | 22.88 | 5.13 |
| | Raw signal min | 5.50 | 5.57 |
| Frequency | PSD mean | 9.23 | 5.96 |
| | PSD var. | 7.28 | $NaN$ |
| | PSD std. | 7.45 | 6.24 |
| | PSD max | 5.18 | $NaN$ |
| | PSD min | 6.05 | 5.95 |
| | Band $0.5-45$ | 3.35 | 9.85 |
| | Band $0.5-3.5$ | 3.52 | 8.48 |
| | Band $3.5-8$ | 5.86 | 7.12 |
| | Band $8-13$ | 15.90 | 6.00 |
| | Band $13-22$ | 7.83 | 6.59 |
| | Band $22-40$ | 14.75 | 14.53 |
| Filter | Order 1 | 4.09 | 3.48 |
| | Order 2 | 3.81 | 4.29 |
| | Order 3 | 3.35 | 9.85 |
| | Order 4 | 2.95 | 10.85 |
| | Order 5 | 3.45 | 1.09 |

Table 4.2: The DBI on which feature selection was based. (Continued from table 4.1)

- Mean of raw signal.
- Maximum of the power spectral density.

The non-linear transform of the PSD maximum value caused the DBI algorithm to try dividing with zero which maked MATLAB return a measure that indicates that the result was *not a number* ($NaN$).

### Interpretation of results

Only signals that corresponded to the Pioneer robot turning right and left were used to obtain the results described here. The primary reason for this was that the results that were obtained here were compared to the results obtained by other researchers that only considered two actions.

It can be seen in table 4.1 that the signal imagery exercise gave a lower DBI than motor imagery. This was very surprising since movement imagery is widely adopted in similar research such as [10] and [55].

Another observation was that the channels that gave the lowest DBI were all confined to the brains frontal lobe, see figure 3.1, page 32. This might have been due to the fact that during the experiment the subjects were asked to maintain a mental activity for six seconds which was difficult to do with the same intensity as in the first, say, second. It is therefore possible that this effort produced more activity in the frontal lobe which is also referred to as the *motor associative cortex* [8] and is responsible for planning and executing movements. The planning was perhaps a bigger factor here than expected which might explain these results.

The time window that gave the lowest DBI was the $0 - 1$ window which was expected to have higher DBI than the $0 - 0.5$ window which, in theory, should contain descriptive EEG components since the burst of RPs was contained within that window, see section 2.4.2. A possible reason for this is that although the RP was present in the first 500 milliseconds, it might have been a response to the cue regardless of the action it indicates. A cue might then have caused a reaction in the brain which was similar for all four cues for the first few hundred milliseconds and not until the subjects associated

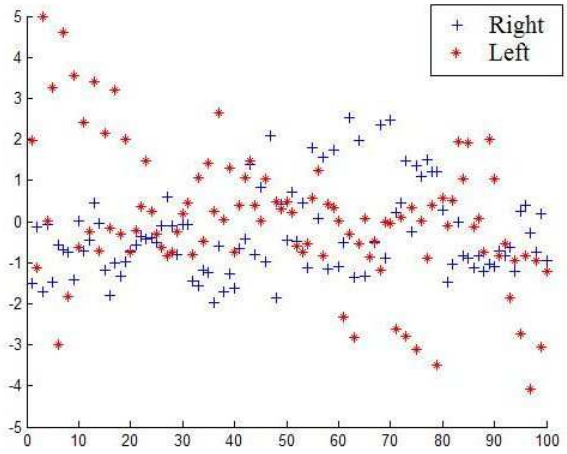



Figure 4.2: A figure of the class overlap between the selected features for right and left.

the cue with an action the RPs became different, which might have taken more than 500 milliseconds.

The mean that was taken over the raw EEG had the least class overlap according to table 4.2. In section 2.4, page 16, it was mentioned that due to the stochastic nature of EEG it is difficult to find a correlation between the shape of the EEG and a specific target. There might, however, have be some measurable correlation between the amplitudes of the EEG and their corresponding target which would make the mean amplitude a descriptive feature.

The DBI for the PSD features were quite similar which might be traceable to the fact that the PSD was distributed over all frequencies that were contained in the signal and a measure such as the mean taken over the whole frequency range may have been a very descriptive feature.

Tables 4.1 and 4.2 also show the there was more class overlap in the non-linearly transformed signals than the original signals, at least for the selected features. This was surprising since non-linearly separable data is more likely to be linearly separable when cast into a higher-dimensional space [15].

The class overlap between right and left hand signal imagery is illustrated in figure 4.2. It is clear that a straight line could not separate these.

## 4.4 Pattern classification

The results described in this section were obtained by applying the classification techniques described in sections 2.5 and 3.4. Confusion matrices of the predictions of each classifier are given as well as their average classification accuracy and their average mean square error. The results were obtained by the MATLAB code in `perceptron.m`, `rbf.m` and `lvq.m` which are given in appendix C.8 to C.10.

The perceptron was initialised with new random weights in each of the three runs using MATLAB's `rands` function. The weights could also be initialised to zero, which is MATLAB's default setting for a perceptron created using `newp`. This was done by including `net.initParam = 0` in the code and resulted in similar performance.

The training was usually suspended after 80 - 100 epochs when the training error and validation error began to deviate. Training continued for all 500 epochs otherwise.

A new RBF network was created in each of the 10 runs and trained and tested with different parts of the data each time. The average number of neurons that were created was around 50.

A new LVQ network was also created 10 times like the RBF network.

The perceptron had peak performance of 79% accuracy and 0.8501 mean square error. The RBF had peak performance of 75% accuracy (100% for right signal imagery) and 1.7950 mean square error while the LVQ network had peak performance of 46% accuracy and 0.3571 mean square error.

#### Interpretation of results

The perceptron exhibited considerably higher classification accuracy than the other two which was surprising since it was the simplest of the three. It is most likely due to the systematic process of choosing features that had the lowest DBI.

Referring back to tables 4.1 and 4.2, one can see that the selected features had a considerably

Perceptron

| | | Actual | |
|---|---|---|---|
| | | Right | Left |
| **Predicted** | Right | 77.78% | 44.45% |
| | Left | 22.22% | 55.56% |

| **Average classification accuracy** | **Average mean square error** |
|---|---|
| 70.2778% | 1.0478 |

RBF network

| | | Actual | |
|---|---|---|---|
| | | Right | Left |
| **Predicted** | Right | 42.50% | 50.00% |
| | Left | 57.50% | 50.00% |

| **Average classification accuracy** | **Average mean square error** |
|---|---|
| 46.25% | 2.2455 |

LVQ network

| | | Actual | |
|---|---|---|---|
| | | Right | Left |
| **Predicted** | Right | 54.29% | 72.86% |
| | Left | 45.71% | 27.14% |

| **Average classification accuracy** | **Average mean square error** |
|---|---|
| 40.71% | 0.4143 |

Table 4.3: The performance of the perceptron, RBF and LVQ networks measured as the average classification accuracy and average mean square error.

high DBI when calculated post non-linear transform. This observation was an indicator that the RBF network would not perform well when used to classify the signals since it uses the same function as was used in the DBI calculations - MATLAB's `radbas` - to transform its inputs. Afterwards, the RBF network performed linear separation of the outputs from the RB layer which would not be readily done according the DBI.

A similar argument holds for the LVQ network which performed clustering on its inputs and keeping in mind that the DBI is in essence a cluster overlap measure one can see that the LVQ networks suffers a similar fate as the RBF network when choosing features based on DBI or a similar measures.

The perceptron and RBF network produced a similar number of false-positives for right and left, i.e., predicting right when the actual action is left, which is more convenient than than the large difference produced by the LVQ network. The reason for this is that a system implemented with the LVQ classifiers would have a great tendency to make the robot turn left when the user was trying to make it turn right. If implemented with the perceptron or RBF network, however, it would have almost equal tendency towards both directions.

# Chapter 5

# Discussion

*This chapter contains a discussion of the pros and cons of the adopted methodology and its application as well as the results.*

## 5.1 Experiment procedure

The Brooker laboratory was a well suited location because of its size which imposed no constraints on the roaming of the Pioneer robot, although the robots movements were programmed to be confined within a 75 centimeter radius. Each of the three experiments proceeded without any major complications and the feedback from the test subjects was mostly positive. There was one aspect, at least, of the experiment protocol which could have been improved to ensure the alertness of the subjects and thus the quality of the EEG. This aspect was the duration of the experiments which were usually between 120 and 180 minutes, a period long enough for people to start loosing their concentration, especially when comfortably seated. The timing constraints on the project and the amount of time required for the setup, did not allow for this improvement.

Another aspect which might be considered was the presence of other students in the laboratory. Although not intended, their activities created disturbances for the subjects, particularly in the latter parts of the experiments.

A possible complication when applying the design described in section 3.5 to solve a practical

problem such as EEG-driven wheelchair control, is that, during the experiment, the subject and the Pioneer robot live in two different coordinate systems. This, however, depends on the effect that visually related EEG have on the classification of the selected features. A measure for this effect might be obtained by conducting a fourth experiment with one of the test subjects and ask them to keep their eyes closed and use a touch or acoustic cues instead.

Exercise one of the experiment was motivated by the use of similar methods conducted by experienced researchers in this field, most notably G. Pfurtscheller *et al.* [57]. The exercise described in that paper were similar to the one used here except it only considers right and left motor imagery and uses acoustic cues as well as visual. They also extended their initial experimental paradigm to include testing sessions where the user would get feedback from the system communicating the results of the classification. It is generally thought that the control of EEG is learnable [16] which may be what they intended. Feedback is also used in [55] and could have been adopted here if given more time.

Exercise two was somewhat motivated by the fact that it is a novel approach and an interesting comparison would be to hold an established technique, such as motor imagery, up against something that had little scientific support. The specifics of this exercise were not encountered elsewhere and the only encounter with an alternative to motor imagery was described in [50] which involved arithmetic operations and is the only publication of an alternative exercise according to [16]. This indicates that choosing from existing alternatives was not a reasonable option since arithmetic operations are not intuitively associated with robot control.

The results in table 4.1, page 59, indicated that the signal imagery would be easier to separate with a linear classifier than the RBF and LVQ networks which is surprising considering the definition of the DBI.

The arguments for the usefulness of this exercise are, however, not built on any scientific foundation since there was no investigation into the physiological aspects of this exercise and thus, difficult to defend.

Considerable time was saved during the implementation of the experiment software since libraries were available for both the SCSI board and the Pioneer robot (ARIA) as well as existing code that implemented a similar experiment.

Timing was a crucial factor in experiments such as these because the EEG that was recorded had to be marked at different places so that relevant signals could be readily extracted and irrelevant signals, recorded whilst the subjects relaxed, could be discarded. Visual inspection indicated that the markers were properly distributed but further investigation would be in order to confirm this.

## 5.2 Signal preprocessing

After the experiment and when the signals had been gathered, they were partitioned so that EEG related to, say, right signal imagery, was all stored in the same file. This made it very easy to work with henceforth. The markers were simply added to the files by including them in an additional row which was not a very elegant solution. A better solution would have been to create separate marker files for each data file during partitioning.

The signals were then filtered with a Butterworth filter which was chosen because of the maximal smoothness of the frequency response, as was described in section 2.3, page 14. The Chebyshev filter was also considered and did not share the flat frequency response of the Butterworth although it has a steeper decline in the stopband for low orders. This difference was not subject to extensive testing, rather it was assumed that the Butterworth would preserve more of the signal over all frequencies while the Chebyshev would preserve more in the vicinity of the stopband. An experiment that would establish which was better in this case would certainly be interesting.

## 5.3 Feature selection

The feature selection was conducted in a scientific way and all features that were used have an associated index which indicates how well a linear classifier could separate them. There are some considerations regarding the applicability of DBI in BCI systems such as the non-linear nature of EEG which is

very unlikely to be linearly separable unless more advanced feature selection methods are used such as the bispectrum of EEG [74]. This indicates that the DBI could be a useful measure when used on statistical properties of the EEG that are obtained with sophisticated techniques. The sophistication of the features that could be considered here was constrained by the amount of time that was available as well as the considerable effort that was used on background research.

In table 4.2, page 60, it can be seen that the frequency band that gave the lowest DBI was the 0.5 - 45Hz broad band. This might be directly traceable to the order of the filter that was used, 3 (12 in effect), since this band contained considerably more frequencies than the others. The effect of the filtering would thus be less in this band than the others and it might have caused more loss of information in the narrower bands which would explain why their DBI was relatively high. There might be other reasons as well but this was very peculiar since much of the related research focuses on the $\alpha$ and $\beta$ bands with similar filter order, e.g., [4], although precedence exists for using broad bands with with good results [57].

The electrodes corresponding to channels $Fp1$, $Fpz$, $Fp2$, $F3$, $Fz$ and $F4$ were all located on top of the frontal lobe which was a very interesting discovery. This might be related to the nature of the exercise which involved signal imagery that might be more related to planning than the execution of movements. It would be interesting to explore how the features would change if motor imagery were used instead. The channels would then be expected to be confined to the motor cortex region (channels $C3$, $Cz$ and $C4$) which have been used as features in, among others, [74].

The first second of each trial had the lowest DBI and was therefore used as a feature. This is probably directly related to the conscious reaction of the subject to the cue, after which the subject tries to maintain the state which might not have produced as descriptive EEG. It was a surprise that the first 500 milliseconds did not contain more descriptive EEG because of the strong RP which is produce during that time after the cue. This window has been used in related work such as [58]. There might even be a slight chance that this was due to inaccurate placement of the markers, a speculation which could not be dismissed without thorough testing.

## 5.4 Pattern classification

The classifiers that were used have different approaches to pattern classification which makes them well suited for this project as it gives three different perspectives into how successful the feature selection was.

The low classification rates are not a testimony of the applicability of classifiers, rather it indicates that low DBI features do not necessarily mean good performance when used with the RBF or LVQ networks.

Remarkable classification accuracies have been achieved with linear classifiers in the past but with different feature selection criteria. For example, [4] reports obtaining 92% classification accuracy. SOM based techniques have been applied before and an acceptable classification rate of 85% was accomplished by [18] using a modified signal space projection classifier. Genetic algorithms have been show to produce results of up to 87% [72] and a modified MLP classifier has given 87% [27] classification accuracy.

The number of neurons used in both the perceptron and the LVQ networks were not determined through thorough testing which should have been the case. Instead, a number of classifiers were trained with a varying number of neurons and the optimal number was approximated according to the results.

## 5.5 The design

The outlines of an EEG-based BCI system that can control a mobile robot was given in section 3.5. These outlines assume that the code that was already available, and had been used in this project, be modified and tailored to the specifications of a real-time system. This would be readily done using the available resources and the fact that the MATLAB code, which does most of the processing, could be compiled into C++ code with MATLAB's `mcc` compiler and included in the robot controller application. Unfortunately, due to timing constraints, this was not completed and thus remains an

assumption.

## 5.6 The report and project experience

This has been a very exciting project, mostly on account of the vast number of previously unfamiliar technologies as well as the theories of neuroscience and signal processing. A lesson has been learned and relates to the extent to which one can establish causal relationships between different parts of a system and its final outcome when proper background research has been conducted.

Many theories have been touched upon, and this report has tried to give as clear and concise account of these as considered feasible.

# Chapter 6

# Project Management

This project began with a written proposal of expected goals and a summary of techniques that would be used. New discoveries and knowledge obtained during the course of the project did, however, shift the focus slightly from an implemented end product to proper research conduct and producing valid reasons for each significant design choice while the context remained the same, i.e., the design of a BCI system. The primary reason for this is that the design could not be justified unless reasons and/or statistics were given for each choice, however successful it turned out to be, and that an implementation of a faulty design is less useful than just the design of a correct system, which could be implemented.

Another reason for this shift in focus was the underestimated time requirements of literature gathering and background research required by the theories of neuroscience and signal processing, both of which the author was not familiar with prior to the project.

A valuable lesson was learned from this which is that a good overview of the problem domain is pivotal for the design of a good project plan.

Below are the original project plan and a discussion on the revised plan.

## 6.1 Original project plan

The original project plan consisted of nine distinct tasks which were the following:

1. *Literature gathering and reading* is an essential part of a research project. This will include gathering literature on the brain's structure and brain waves, EEG tools, filters, feature extraction, classification and BCI systems in general and their application. This task should be started in the third week of June and will run for two weeks.

   (a) Finding books.

   (b) Finding related work.

   (c) Finding scientific articles.

   (d) Glossary of findings.

2. *Revision of project plan* may or may not be required. This depends on the literature and any new techniques which may be encountered. This task will run in parallel with the previous one but will only last two days. This task defines the first milestone when completed.

   (a) Revise problem domain.

   (b) Revise existing technologies.

   (c) Create a new project plan if needed.

3. *Analysis and documentation* will be done during the first two weeks of July. Once it has been completed, then the second milestone has been accomplished.

   (a) Analyse requirements.

   (b) Analyse problem domain.

   (c) Analyse applicability of the techniques with regards to the time given.

   (d) Document the analysis.

4. *System design and documentation* describe the choices which have been made based on the information established in task 3. It will run for two weeks and the third milestone will be reached once it has been completed.

   (a) Choose EEG system.

   (b) Choose a filter.

   (c) Choose feature extraction algorithms.

   (d) Choose classifiers.

   (e) Document the design and the reasons for the choices.

5. *Implementation* will run for two weeks where the first week runs in parallel with task 4. It contains the implementation of the BCI system. Milestone 4 is reached after its completion.

   (a) Implement the design described in task 4.

6. *Testing and result gathering* is the largest and most significant part of the project plan as it contains the actual results of the whole project. It runs for three weeks, two of which are parallel with task 5, and defines milestone 5.

   (a) Implement tests.

   (b) Perform tests.

   (c) Gather results, i.e., statistics, graphs and illustrations.

7. *Implementation and testing documented* runs in parallel with the last week of task 6 and defines milestone 6 upon its completion.

   (a) Document the implementation from task 5.

   (b) Document the testing from task 6.

8. *Revision of results and documentation* will run for one week after everything else has been finished. This task can also accommodate other tasks if they take more time then anticipated. It includes milestone 7 which will be reached upon its completion.

   (a) Find the cause for any unexpected results and document.

(b) Find any inconsistencies in the documentation.

9. *Preparation for presentation* will run for two weeks once everything else has been completed.

(a) Produce slides.

The Gantt diagram that describes this is given in appendix D.

Project management was a task that would be performed concurrently with the other tasks throughout the project.

## 6.2 Revised project plan

There were some complications as was described earlier. The primary cause of this was the underestimation of the amount of background research that needed to be conducted. The background research became both more extensive and time consuming than expected and caused the goals of the project to be reconsidered. Once this was apparent, the project plan was revised and structured in a way that background research and literature gathering would run in parallel with implementation and testing of the different parts of the project. Results were gathered underway and documented when new discoveries were made. This allowed for more flexibility regarding the applied techniques as there was no predefined design that needed to be implemented. Rather, techniques that seemed to give reasonable results were tested and tried when a justification for their application was established.

The revised project plan included the same tasks as the original but they were ordered differently.

Milestones one and two had already passed when the plan was revised so the remaining five milestones were redefined as follows:

- *Milestone* 3: Complete revision of project plan. Gather literature and background research.
- *Milestone* 4: Experiment protocol defined and experiment code finished.
- *Milestone* 5: Experiments completed and data gathered and preprocessed.

- *Milestone* 6: Use selected techniques to select and extract features, try them using three different classifiers.
- *Milestone* 7: Evaluate and document findings.

## 6.3 Project tracking

A internet diary was kept throughout the project which is shown in appendix D. There it can be seen what milestones were reached as well as reasons for any delays.

# Chapter 7

# Conclusions

This report was a dissertation of a research project that was conducted into the design of BCI systems. Many theories and techniques were described to an extent that reflects the authors initial unfamiliarity with the problem domain.

One conclusion that can be drawn from the project was that background research into underlying theories and techniques was a very important foundation on which evaluations of the results were based. The reasons for this were, firstly, that the results of this research could not be compared with the results of related research unless the effect of each applied technique in both of them was fully understood. Secondly, the results could not have been discussed at a professional level without a solid knowledge of the possible causes and complications. The problem domain was BCI systems which includes theories and techniques from many directions such as neuroscience, signal processing and artificial intelligence, which made this project particularly demanding.

A number of experiments were conducted and reported, and the results from these were used to justify design decisions that were made.

Features from three different domains were considered which were the spatial, time and frequency domains, and features were selected based on a statistical measure, the DBI, which measured how

separable the classes were with a linear classifier.

The classification accuracy of the classifiers was varying and an average of 70% was established for the perceptron. Related research have shown that linear classifiers can have acceptable performance when used on carefully selected features. There was an indication that the DBI was not a reasonable choice in this case when more advanced techniques were available. However, the feature selection remains statistically justified which conforms to proper professional conduct which was one of the main goals of this project.

A surprising discovery was made which included comparing the widely adopted motor imagery exercise to the novel signal imagery exercise where the latter was shown to give a lower DBI.

# Chapter 8

# Future work

There are some speculations that remain regarding the techniques described in the report and if given a few more months, the following would be interesting to consider.

Firstly, regarding the experiments, more of them could be conducted where subjects would be asked to have their eyes closed and acoustic cues could be used to communicate the direction in which the Pioneer would move. Comparing the signals obtained during such experiments to the previously obtained signals would give some information on how the visual aspect affects EEG. Furthermore, the experiments could be shortened to ensure the concentration of the subjects throughout the experiments.

Feedback to the subjects could also be included in the experiments. This is thought to have the effect that the subjects are trained as well as the classifiers, and will ultimately have better control over the informative bursts of EEG that are otherwise only found in a relatively short segment recorded after the cue.

A research into the physiological aspects of exercise two would also be very interesting based on its low DBI compared to exercise one.

A more descriptive feature than the mean of the PSD might have been obtained by finding the principal components for the PSD of each of the exercises to find a specific frequency that best separated them. This could then be used to further examine the difference between the two.

Comparing the features used here with a variety of popular features such as the combination of

exercise one, channels $C3$, $Cz$ and $C4$ over the PMA and the window between 0s and 0.5s, with AR or AAR parameters.

Timing is very important in signal acquisition for BCI research and a research into the duration of each operation of the implemented experiment paradigm, as well as the delay between the transmitting of a command by the application and the actual execution of it by the Pioneer robot is believed to be a worth while venture.

The difference of applying the Butterworth and Chebyshev filters could be established with experiments.

The classification part could be extended with research into the optimal values of structural and learning parameters such as the learning rate, epochs, etc. It could also be extended to include hybrid systems such as comity machines where a number of classifiers vote for the final output.

Lastly, elaborating on the design and implementing the real-time application. The application could then be extended to apply a technique described in [7] that allows the system to detect when the user intends to control the device, and is idle otherwise.

# Bibliography


[1] M. Abeles. *Local Cortical Circuits: An Electrophysiological Study.* Springer-Verlag, 1982.

[2] D. J. Amit. *Modeling Brain Function: The World of attractor neural networks.* Cambridge University Press, 1989.

[3] C. Babiloni, F. Babiloni, F. Carducci, F. Concotti, C. Percio, M. Hallett, D. V. Moretti, G. L. Romani, and P. M. Rossini. High resolution eeg of sensorimotor brain functions: mapping erps or mu erd? *Advances in Clinical Neurophysiology*, 54:365 – 371, 2002.

[4] F. Babiloni, F. Cincotti, L. Lazzarini, J. Millán, J. Mouriño, M. Varsta, J. Heikkonen, L. Bianchi, and M. G. Marciani. Linear classification of low-resolution eeg patterns produced by imagined hand movements. *IEEE Transactions on Rehabilitation Engineering*, 8(2):186 – 188, 2000.

[5] G. E. Birch, S. G. Mason, and J. F. Borisoff. Current trends in brain-computer interface research at the neil squire foundation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):123 – 126, 2003.

[6] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[7] J. F. Borisoff, S. G. Mason, A. Bashashati, and G. E. Birch. Brain-computer interface design for asynchronous control applications: Improvements to the lf-asd asynchronous brain switch. *IEEE Transactions on Biomedical Engineering*, 51(6):985 – 992, 2004.

[8] N. R. Carlson. *Physiology of behaviour.* Allyn and Bacon, 2001.

[9] P. S. Churchland. *Neurophilosophy: toward a unified science of the mind-brain.* MIT Press, 1986.

[10] F. Cincotti, D. Mattia, C. Babiloni, F. Carducci, S. Salinari, L. Bianchi, M. G. Marciani, and F. Babiloni. The use of eeg modifications due to motor imagery for brain-computer interfaces. *IEEE Transactions on Neural Systems and REhabilitation Engineering*, 11(2):131 – 133, 2003.

[11] Personal communication of Dr. Francisco Sepulveda.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 2001.

[13] E. J. X. Costa and E. F. Cabral Jr. Eeg-based discrimination between imagination of left and right hand movements using adaptive gaussian representation. *Medical Engineering and Physics*, 22:345 – 348, 2000.

[14] E. J. X. Costa and E. F. Cabral Jr. Eeg-based discrimination between imagination of left and right hand movements using adaptive gaussian representation. *Medical Engineering & Physics*, 22:345 – 348, 2000.

[15] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326 – 334, 1965.

[16] E. A. Curran and M. J. Stokes. Learning to control brain activity: A review of the production and control of eeg components for driving brain-computer interface (bci) systems. *Brain and Cognition*, 51:326 – 336, 2003.

[17] D. Davies and D. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 1:224 – 227, 1979.

[18] J. del R. Rillán, J. Mouriño, M. Franzé, F. Cincotti, M. Varsta, J. Keikkonen, and F. Babiloni. A local neural classifier for the recognition of eeg patterns associated to mental tasks. *IEEE Transactions on Neural Networks*, 13(3):678 – 686, 2002.

[19] E. Donchin, K. M. Spencer, and R. Wijesinghe. The mental prosthesis: assessing the speed of a p300-based brain-computer interface. *IEEE Transaction on Rehabilitation Engineering*, 8:174 – 179, 2000.

[20] G. M. Edelman and V. B. Mountcastle. *The Mindful Brain.* MIT Press, 1982.

[21] Wikipedia: The Free Encyclopedia. *http://www.wikipedia.org*, Last checked on the 16th of September 2005.

[22] L. A. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70:510 – 523, 1988.

[23] T. C. Ferre, P. Luu, G. S. Russell, and D. M. Tucker. Scalp electrode impedance, infection risk, and eeg data quality. *Clinical Neurophysiology*, 112:536 – 544, 2001.

[24] M. A. Goodale and A. D. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15:20 – 25, 1992.

[25] C. Guger, H. Ramoser, and G. Pfurtscheller. Real-time eeg analysis with subject-specific spatial patterns for a brain-computer interface (bci). *IEEE Transaction on Rehabilitation Engineering*, 8(4):447 – 456, 2000.

[26] Jesper H. Report of comittee on methods of clinical exam in eeg. *Electroencephalography and Clinical Neurophysiology*, 10:370 – 375, 1958.

[27] E. Haselsteiner and G. Pfurtscheller. Using time-dependent neural networks for eeg classification. *IEEE Transactions on Rehabilitation Engineering*, 8:457 – 463, 2000.

[28] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[29] D. O. Hebb. *The Organization of Behaviour: A Neuropsychological Theory*. Wiley-Interscience, 1949.

[30] MATLAB help. *http://www.mathworks.com/access/helpdesk/help/helpdesk.html*, Last checked on the 16th of September 2005.

[31] C. Kyoung Ho and M. Sasaki. Brain-wave bio potentials based mobile robot control: Wavelet-neural network pattern recognition approach. Technical report, Department of Mechanical & Systems Engineering, Gifu University, Japan, 2001.

[32] J. R. Hodges, J. Spatt, and K. Patterson. What and how: Evidence for the dissociation of object knowledge and mechanical problem solving skills in the human brain. *Reoceedings of the National Academy of Science*, 96(16):9444 – 9448, 1999.

[33] T. W. James, J. Culham, G. K. Humphrey, A. D. Milner, and M. A. Goodale. Ventral occipital lesions impair object recognition but not object directed grasping: an fmri study. *Brain*, 126:2463 – 2475, 2003.

[34] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. McGraw-Hill Medical, 2000.

[35] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59 – 69, 1982.

[36] A. Kostov and M. Polak. Parallel man-machine training in development of eeg-based cursor control. *IEEE Transactions on Rehabilitation Engineering*, 8(2):203 – 205, 2000.

[37] B. Libet, C. A. Gleason, E. W. Wright, and D. K. Pearl. Time of conscious intention to act in relation to onset of cerebral activity (rediness-potential). the unconscious initiation of a freely voluntary act. *Brain*, 106:623 – 642, 1983.

[38] M. A. Paradiso M. F. Bear, B. W. Connors. *Neuroscience - Exploring the Brain*. Williams & Wilkins, 1996.

[39] The MathWorks. *http://www.mathworks.com*, Last checked on the 16th of September 2005.

[40] W. S. McCulloch and W. H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115 – 133, 1943.

[41] T. M. Michell. *Machine Learning.* McGraw-Hill, 1997.

[42] M. Middendorf, G. McMillan, G. Calhoun, and K. S. Jones. Brain-computer interfaces based on the steady-state visual-evoked response. *IEEE Transactions on Rehabilitation Engineering*, 8(2):211 – 214, 2000.

[43] A. D. Milner and M. A. Goodale. *The visual brain in action.* Oxford University Press, 1995.

[44] Morphonix. *http://www.morphonix.com/software/education/science/brain/game/specimens/cerebral_ cortex_ lobes.gif*, Last checked on the 16th of September 2005.

[45] K. Müller, C. W. Anderson, and G. E. Birch. Linear and nonlinear methods for brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):165 – 169, 2003.

[46] J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg. Classification of movementrelated eeg in a memorized delay task experiment. *Clinical Neurophysiology*, 111:1353–1365, 2000.

[47] B. Obermaier, C. Munteanu, A. Rosa, and G. Pfurtscheller. Asymmetric hemisphere modeling in an offline brain-computer interface. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 31(4):536 – 540, 2001.

[48] University of Bristol. *http://137.222.110.150/calnet/sens/image/somatotopic%20arrangement%20in%20sensory%20cortex-human.jpg*, Last checked on the 16th of September 2005.

[49] W. D. Penny, S. J. Roberts, E. A. Curran, and M. J. Stokes. Eeg-based communication: A pattern recognition approach. *IEEE Transactions on Rehabilitation Engineering*, 8(2):214 – 215, 2000.

[50] W. D. Penny, S. J. Roberts, E. A. Curran, and M. J. Stokes. Eeg-based communication: A pattern recognition approach. *IEEE Transactions on Rehabilitation Engineering*, 8:214 – 215, 2000.

[51] B. O. Peters, G. Pfurtscheller, and H. Flyvbjerg. Mining multi-channel eeg for its information content: an ann-based method for a brain-computer interface. *Neural Networks*, 11:1429 – 1433, 1998.

[52] G. Pfurtscheller. Eeg event-related desynchronization (erd) and event-related synchonization (ers). *Electroenceph.: Basic princ. clin. appl.*, pages 958 – 967, 1999.

[53] G. Pfurtscheller, D. Flotzinger, and J. Kalcher. Brain-computer interface - a new communication device for handicapped persons. *Journal of Microcomputer Applications*, 16:293 – 299, 1993.

[54] G. Pfurtscheller, J. Kalcher, Ch. Neuper, D. Flotzinger, and M. Pregenzer. On-line eeg classification during externally-paced hand movements using a neural network-based classifier. *Electroencephalography and clinical Neurophysiology*, 9:416 – 425, 1996.

[55] G. Pfurtscheller and C. Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123 – 1134, 2001.

[56] G. Pfurtscheller, C. Neuper, C. Guger, W. Harkam, H. Ramoser, A. Schlögl, B. Obermaier, and M. Pregenzer. Current trends in graz brain-computer interface (bci) research. *IEEE Transaction on Rehabilitation Engineering*, 8(2):216 – 219, 2000.

[57] G. Pfurtscheller, C. Neuper, A. Schlögl, and K. Lugger. Separability of eeg signals recorded during right and left motor imagery using adaptive autoregressive parameters. *IEEE Transactions on Rehabilitation Engineering*, 6(3):316 – 325, 1998.

[58] J. A. Pineda, B. Z. Allison, and A. Vankov. The effects of self-movement, observation, and imagination of $\mu$ rhythms and readiness potentials (rp's): Towards a brain-computer interface (bci). *IEEE Transactions on Rehabilitation Engineering*, 8(2):219 – 222, 2000.

[59] M. Pregenzer and G. Pfurtscheller. Frequency component selection for an eeg-based brain to computer interface. *IEEE Transactions on Rehabilitation Engineering*, 7(4):413 – 419, 1999.

[60] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing: Principles, Algorithmsl, and Applications*. Prentice-Hall, 1996.

[61] Allied Products. *http://www.biof.com/mindset.html*, Last checked on the 16th of September 2005.

[62] H. Ramoser, J. M üller Gerking, and G. Pfurtscheller. Otimal spatial filtering of single trial eeg during imagined hand movement. *IEEE Transaction on Rehabilitation Engineering*, 8(4):441 – 446, 2000.

[63] ActiveMedia Robotics. *http://www.activmedia.com*, Last checked on the 16th of September 2005.

[64] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 6:386 – 408, 1958.

[65] F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, 1962.

[66] A. Schlögl, G. Müller, C. Neuper, G. Krausz, B. Graimann, and G. Pfurtscheller. Adaptive autoregressive parameters used in bci research. Technical report, Institute for Biomedical Engineering, University of Technology Graz, Austria.

[67] A. Schloegl, K. Lugger, and G. Pfurtscheller. Using adaptive autoregressive parameters for a brain-computer interface experiment. *Proceedings - IEEE/EMBS 19th International Conference*, Oct. 30 - Nov. 2:1533 – 1535, 1997.

[68] BrainMaster Technologies. *http://www.brainmaster.com/generalinfo/electrodeuse /eegbands/*1020/1020*S*02*.gif*, Last checked on the 16th of September 2005.

[69] A. B. Williams and F. J. Taylor. *Electronic Filter Design Handbook*. McGraw-Hill, 1995.

[70] S. B. Wilson, C. A. Turner, R. G. Emerson, and M. L. Scheuer. Spike detection ii: automatic, perception-based detection and clusterin. *Clinical Neurophysiology*, 110:404 – 411, 1999.

[71] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113:767 – 791, 2002.

[72] E. Yom-Tov and G. F. Inbar. Feature selection for the classification of movements from single movement-related potentials. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(3):170 – 177, 2002.

[73] S. K. Yoo, S. H. Kim, N. H. Kim, Y. T. Kang, K. M. Kim, S. H. Bae, and M. W. Vannier. Design of a pc-based multimedia telemedicine system for brain function teleconsultation. *International Journal of Medical Informatics*, 61:217 – 227, 2001.

[74] S. Zhou, J. Q. Gan, and F. Sepulveda. Using higher-order statistics from eeg signals for developing brain-computer interface (bci) systems. Technical report, Department of Computer Science, University of Essex, The UK, 2004.

# Part I

# Appendix

# Appendix A

# Notation

*This chapter of the appendix contains a summary of abbreviations, notations, fonts and symbols used throughout the report.*

| Notation | Description |
|---|---|
| $\overline{x}$ | $x$ is a $N$ element vector $x = (x_1, \ldots, x_N)$ |
| $N$s | $N$ seconds |
| $A^T$ | The transpose of array $A$ |
| Exercise | One of the two mental exercises used in the experiment, motor imagery or signal imagery |
| Neural network | Biological neural network unless stated otherwise |

Table A.1:

| Font | Used for |
|---|---|
| *LaTeX emphasised font* | First mention of an important term |
| LaTeX *mathfont* | Variables. |
| `LaTeX typewriter font` | Function, algorithm and container names |
| **LaTeX bold font** | Table row and column names |

Table A.2:

| Symbol | Meaning |
|---|---|
| # | Number of instances |
| \| | Logical *or* |

Table A.3:

| Abbreviation | Full name | Introduced on page # |
|---|---|---|
| BCI | Brain-computer interface | 1 |
| PMA | Primary motor area | 10 |
| EEG | Electroencephalogram | 13 |
| RP | Readiness potential | 18 |
| DFT | Discrete fourier transform | 19 |
| FFT | Fast fourier transformation | 20 |
| PSD | Power spectral density | 21 |
| AR | Autoregressive model | 22 |
| AAR | Adaptive autoregressive model | 22 |
| PCA | Principal component analysis | 23 |
| DBI | Davies-Bouldin index or indices | 23 |
| MSE | Mean square error | 48 |
| MLP | Multilayer perceptron network | 26 |
| RB(F) | Radial basis (function network) | 26 |
| K-NN | K-nearest neighbour clustering | 27 |
| LVQ | Learning vector quantisation network | 28 |
| SOM | Self-organising map algorithm | 28 |
| SCSI | Small computer system interface | 33 |
| ARIA | ActiveMedia robotics interface for application | 33 |

Table A.4:

# Appendix B

# Voluntary consent form

*This form was created to make sure that individuals from outside the Essex University Department of Computer Science would be aware of the risks involved and willing to participate in spite of the possibility of a minor injury.*

## B.1 Voluntary consent form

UNIVERSITY OF ESSEX COMPUTER SCIENCE DEPARTMENT
BRAIN-COMPUTER INTERFACES LABORATORY
VOLUNTEER CONSENT FORM
Student: Hinrik J. Atlason
Supervisor: Dr. Francisco Sepulveda
Brief outline of project The purpose of the project is to use EEG signals for navigating a robot using software developed for these purposes. The EEG signals will also be analysed to find movement event related signatures. For doing this we will develop signal processing software, feature extraction and a classifier.
[The experiment will consist of two parts, in the first one the subject will be asked to visualise the execution of four distinct movements whose type will be indicated by lights on the robot and the movement of the robot. In the second one the subject will be asked to imagine signals travelling from its brain to the corresponding limb which is, again, indicated by lights on the robot and the movement of the robot. Each part consists of four trials, each lasting for four minutes. Rest will be according to the subject's wishes.]
Refreshments (water/juice) and rest periods will be provided to the volunteer subject as and when he/she wishes.
Outline of procedures to be used: An EEG cap will be placed on the subject's head. Electrode gel will be placed into cap orifices by means of a disposable blunt needle. The subject will be seating on a comfortable chair while brain activity signals are recorded for a number of situations concerning guided voluntary movement intentions. The subject will be in front of a computer. Abrasion of the skin with the blunt needle may be necessary if the impedance between skin and electrode is too high (i.e., above 5k ).
Possible Hazards: Improper abrasion with the blunt needle may cause skin damage. Abrasion procedures are to be done according to the instructions provided EEG equipment manual and under supervision by Dr Sepulveda.

Possible discomfort and distress: Although not usual, discomfort and subsequent distress may result from the use of the head cap. Usually, discomfort subdues after about 15-30 min of cap placement. Prolonged use of the equipment (several hours to a day) may cause discomfort as well.

Experience in experiments involving potential hazard, discomfort or distress. Dr. Sepulveda, who is going to be present in all experiments, has had experience with similar experimental conditions and EEG equipment. Further, Dr. Sepulveda has 15 years of experience in other areas of electrophysiological experimentation with humans under potentially hazardous and distressful situations. He has been trained to deal with these situations both involving normal and disabled subjects.

Electrical / electronic equipment will be connected to the subject. The equipment to be used has been approved for use with humans beings. All equipment is electrically isolated from the subject.

Ethical Clearance Use of EEG equipment on human subjects has been approved by the University of Essex's Ethical Committee.

The identity of all volunteers will be kept anonymous All volunteers can abandon an experiment at any time

I have read the complete document and I certify that I am aware of the potential hazards involved and that I accept to be a volunteer.

_____ _____

Volunteer's name Date

Volunteer's signature

# Appendix C

# Source code

*Source code that was mentioned in the report is given here. The remaining source code can be found on the CD that accompanies this report.*

## C.1 experiment.cpp

```
// Most of this code was made by F. Sepulveda.
// Hinrik J. Atlason modified it.
#include <aria.h>
#include <windows.h>
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <io.h>
#include <cstdio>
#include "stdafx.h"
#include "scsi.h"
long st = 0; long et = 0; int bn = 0; unsigned long
samplmark[500]; char z = 0; FILE *file; FILE *timefile;
class MyAcq : public CSCSI {
    virtual void SCSI_OnMessage(LPCSTR msg) {
        MessageBox(NULL,msg, "Error", MB_OK);
    }
    virtual void SCSI_OnForcedAbort(){}
    virtual void SCSI_OnShowBufferStatus(int bytecount){}
    virtual void SCSI_OnDispatchData(BYTE *data, int num, bool first);
public:
};
void MyAcq::SCSI_OnDispatchData(BYTE *data, int num, bool first) {
    if(!ferror(file))
    {
        fwrite(data, num, sizeof(char), file);
```

```
    }
}
// critical variable definitions:
    #define true 1
    #define false 0
    #define CONX 80                                  // console size
    #define CONY 25
    #define FOREGROUND_BLUE                0x0001
    #define FOREGROUND_GREEN               0x0002
    #define FOREGROUND_RED                 0x0004
    #define FOREGROUND_INTENSITY           0x0008
    #define BACKGROUND_BLUE                0x0010
    #define BACKGROUND_GREEN               0x0020
    #define BACKGROUND_RED                 0x0040
    #define BACKGROUND_INTENSITY           0x0080
    #define BACKGROUND_WHITE        (WORD) 0x00f0
    #define BACKGROUND_CYAN         (WORD) 0x0030
    #define FOREGROUND_YELLOW       (WORD) 0x0006
    #define FOREGROUND_CYAN         (WORD) 0x0003
    #define FOREGROUND_WHITE        (WORD) 0x0007
    #define n1 40 // was 80
// general variables
    int                 i,ii,iii,j,k,m;
    int                 lp_num;                      // lp_num = 1 when l<0
    char                choice;                      // char for menu choices
    char                cyclefile[20],filename[40]; // output file name root
    unsigned long       bw;                          // console variable: bytes
                                                     // written per call
    char                fillchar=' ';                // char for cleaning video
                                                     // buffer
    HANDLE              hdout,hdin;                  // console handlers
    COORD               xycoord;                     // structured variable for
                                                     // cursor position
    FILE    *datafile,*network,*fwd_file,*fp,*filein;
    DWORD cWritten;
    const char          *seq[n1]=
    {"RIGHT ARM","LEFT ARM","RIGHT ARM","FEET","MOUTH",
    "RIGHT ARM","LEFT ARM","MOUTH","FEET","FEET","MOUTH",
    "RIGHT ARM","LEFT ARM","MOUTH","FEET","FEET","MOUTH",
    "LEFT ARM","RIGHT ARM","LEFT ARM","RIGHT ARM","FEET","MOUTH",
    "RIGHT ARM","LEFT ARM","MOUTH","FEET","FEET","MOUTH",
    "LEFT ARM","RIGHT ARM","LEFT ARM","RIGHT ARM","FEET","MOUTH",
    "RIGHT ARM","LEFT ARM","MOUTH","FEET","LEFT ARM"};
    MyAcq Acq;
    ArRobot *robot;
    ArGripper *gripper;
// ************************************************
// *****************    SETUP  ********************
// ************************************************
// File1: 1a_movement & 1a_movement_mark
// File2: 1b_movement & 1b_movement_mark
// File3: 2a_movement & 2a_movement_mark
// File4: 2b_movement & 2b_movement_mark
// File5: 1a_signal & 1a_signalt_mark
// File6: 1b_signal & 1b_signal_mark
// File7: 2a_signal & 2a_signal_mark
// File8: 2b_signal & 2b_signal_mark
    // This should be changed to 20 and 39 for sequence 2.
    int start=0;
```

```
    int stop=19;
    int count=start-1;
/*****************************************************************/
//                   FUNCTION DECLARATIONS
/*****************************************************************/
_inline void gotoxy(short a, short b);
_inline void clrscr(void);
_inline void mmenu(void);
_inline void logo(void);
_inline void get_option(void);
_inline void browse_any_file(void);
void runrobot(void);
ArGlobalFunctor runrobotCB(&runrobot);
void blinkAll(void);
void cue(void);
/*****************************************************************/
//                   PROGRAM ROUTINES
/*****************************************************************/
/*****************************************************************/
void blinkAll() {
    robot->com2Bytes(ArCommands::DIGOUT,0x01,0x01); // TURNS RIGHT LIGHT ON!!!
    robot->com2Bytes(ArCommands::DIGOUT,0x04,0x04); // TURNS LEFT LIGHT ON!!!
    robot->com2Bytes(ArCommands::DIGOUT,0x02,0x02); // TURNS FRONT LIGHT ON!!!
    robot->com2Bytes(ArCommands::DIGOUT,0x19,0x09); // TURNS BACK LIGHT ON!!!
    robot->com2Bytes(ArCommands::DIGOUT,0x01,0x02); // TURNS RIGHT LIGHT OFF!!
    robot->com2Bytes(ArCommands::DIGOUT,0x04,0x09); // TURNS LEFT LIGHT OFF!!
    robot->com2Bytes(ArCommands::DIGOUT,0x02,0x09); // TURNS FRONT LIGHT OFF!!
    robot->com2Bytes(ArCommands::DIGOUT,0x19,0x01); // TURNS BACK LIGHT OFF!!
    robot->com2Bytes(ArCommands::DIGOUT,0x01,0x02); // TURN RIGHT LIGHT OFF AGAIN!!
}
void cue() {
    if((count+1)!=(n1))
    {
        if(seq[count+1]=="RIGHT ARM")
        {
            robot->com2Bytes(ArCommands::DIGOUT,0x01,0x01); // TURNS RIGHT LIGHT ON!!!
            robot->com2Bytes(ArCommands::DIGOUT,0x01,0x02); // TURNS FRONT LIGHT OFF!!
        }
        else if(seq[count+1]=="LEFT ARM")
        {
            robot->com2Bytes(ArCommands::DIGOUT,0x04,0x04); // TURNS LEFT LIGHT ON!!!
            robot->com2Bytes(ArCommands::DIGOUT,0x04,0x09); // TURNS LEFT LIGHT OFF!!
        }
        else if(seq[count+1]=="FEET")
        {
            robot->com2Bytes(ArCommands::DIGOUT,0x02,0x02); // TURNS FRONT LIGHT ON!!!
            robot->com2Bytes(ArCommands::DIGOUT,0x02,0x09); // TURNS FRONT LIGHT OFF!!
        }
        else // MOUTH
        {
            robot->com2Bytes(ArCommands::DIGOUT,0x19,0x08); //TURNS BACK LIGHT ON!!!
            robot->com2Bytes(ArCommands::DIGOUT,0x19,0x10); //TURNS BACK LIGHT OFF!!
        }
    }
} void runrobot(void) {
    if(count>=start)
```

```
{
    gotoxy(36,19);cprintf("%6s",seq[count]);gotoxy(0,0);
    samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
    bn++;
    Sleep(1000);
    samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
    bn++;
    Sleep(1000);
    samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
    bn++;
    Sleep(1000);
    samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
    blinkAll();
    bn++;
    gotoxy(36,19);_cputs("             ");gotoxy(0,0);
    Sleep(6000);
}
if(stop>count)
{
    if( seq[count+1]=="RIGHT ARM" )
    {
        cue();
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
        Sleep(1000);
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
        Sleep(1000);
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
        Sleep(1000);
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
        robot -> setVel2(100,-100);
    }
    else if( seq[count+1]=="LEFT ARM" )
    {
        cue();
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
        Sleep(1000);
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
        Sleep(1000);
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
        Sleep(1000);
        robot -> setVel2(-100,100);
    }
    else if( seq[count+1]=="FEET" )
    {
        cue();
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
        Sleep(1000);
        samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
        bn++;
```

```
            Sleep(1000);
            samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
            bn++;
            Sleep(1000);
            robot -> setVel2(100,100);
        }
        else // MOUTH
        {
            cue();
            samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
            bn++;
            Sleep(1000);
            samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
            bn++;
            Sleep(1000);
            samplmark[bn] = Acq.SCSI_GetCurrentSampleNum();
            bn++;
            Sleep(1000);
            robot -> setVel2(-100,-100);
        }
    }
    count++;
    if((count>stop))
    {
        robot->lock();
        robot->disconnect();
        robot->unlock();
        Aria::shutdown();
        gotoxy(36,19);_cputs("             ");
        gotoxy(36,23);printf("  END   \n");
    }
}
int run_sequence(void) {
    char s[20];
    int r = 0;
    clock_t t1, t2;
    double  dt;
    gotoxy(30,21);
    printf("Output file: ");
    scanf("%s", s);
    file = fopen(s, "wb");
    bn=0;
    gotoxy(30,21);
    SetConsoleTextAttribute(hdout,FOREGROUND_BLUE|FOREGROUND_WHITE|BACKGROUND_BLUE);
    printf( "                                    ");
    gotoxy(25,17);
    SetConsoleTextAttribute(hdout,FOREGROUND_BLUE|FOREGROUND_WHITE|BACKGROUND_RED);
    blinkAll();
    blinkAll();
    Sleep(2000);
    gotoxy(25,17);
     _cputs("ÉÍÍÍÍÍÍÍÍÍ          ÍÍÍÍÍÍÍÍÍ»");gotoxy(25,18);
     _cputs("º            GET            º");gotoxy(25,19);
     _cputs("º           READY           º");gotoxy(25,20);
     _cputs("ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼");
     gotoxy(0,0);
```

```
     if(!Acq.SCSI_Init()) {
        //printf("Init Error\n");
        //return 1;
    }
    Acq.SCSI_SetSampleRate(SAMPLES_256SPS);
    gotoxy(25,18);
    _cputs("°                          °");gotoxy(25,19);
    _cputs("°                          °");gotoxy(0,0);
    choice='z';
    i=0;
    t1=clock();
    Acq.SCSI_StartSampling();
    robot->run(true);
    t2=clock();
    Acq.SCSI_StopSampling();
    fclose(file);
    gotoxy(28,19);
    printf("Marker file:  ");
    scanf("%s", s);
    timefile = fopen(s, "w");
    for(int ij = 0; ij<bn; ij++)
        fprintf (timefile, "%i\n",samplmark[ij]);
    fclose(timefile);
    dt = (double)(t2 - t1) / CLOCKS_PER_SEC;
    gotoxy(30,23);
    printf( "%8.4f seconds\n", dt );
    gotoxy(28,19);_cputs("       Stop/Esc       ");
    choice='z'; while((choice!=27)) choice=_getch();
    gotoxy(30,23);
    SetConsoleTextAttribute(hdout,FOREGROUND_BLUE|FOREGROUND_WHITE|BACKGROUND_BLUE);
    printf( "                ");
    return 1;
}
/*************************************************************************/
/*************************************************************************/
void examine_ASCII_files(void) {
  SetConsoleTextAttribute(hdout,FOREGROUND_BLUE|FOREGROUND_WHITE|BACKGROUND_BLUE);
  mmenu();
  gotoxy(25,11);
  cputs("2 - Examine ASCII Files ÍÍÍÍ>");
  gotoxy(55,10);
  cputs("ÉÍÍÍÍ Check  Files ÍÍÍÍÍ»");gotoxy(55,11);
  cputs("°                       °");gotoxy(55,12);
  cputs("° 1 - Browse C program  °");gotoxy(55,13);
  cputs("° 2 - Browse any file   °");gotoxy(55,14);
  cputs("° 3 - Back to Main      °");gotoxy(55,15);
  cputs("°                       °");gotoxy(55,16);
  cputs("ÈÍÍÍÍÍÍ Choice    ÍÍÍÍÍ¼");
  smenu:
    gotoxy(71,16);cputs("?");
    gotoxy(71,16);
    choice='z';
    while(choice<49 || choice>51) choice=getch();
    gotoxy(71,16);
```

```
    cprintf("%c",choice);
    switch(choice)
      {
        case 49:
         {
          gotoxy(61,20);cputs("   Back");gotoxy(61,20);
          system("browse.com exp01_wrist.cpp");
          Sleep(500);
          gotoxy(61,20);
          cputs("                 ");
          break;
         }
        case 50:
         {
          browse_any_file();
          break;
         }
        case 51: goto toMain;
      }
    gotoxy(46,2);
  goto smenu;
  toMain:clrscr();
}
//#########################################################################
//#########################################################################
//                          MAIN PROGRAM
//#########################################################################
//#########################################################################
int main(void)//(int argc, char* argv[]) //void _cdecl main() {
    int ret;
    std::string str;
    // the connection for Remote Host and Simulator
    //ArTcpConnection con;
    // the connection through Serial Link to the robot
    ArSerialConnection con;
    bool useSim=false;
    robot = new ArRobot;
    gripper = new ArGripper(robot);
    // mandatory init
    Aria::init();
    // open the connection, if this fails exit
    if ((ret = con.open()) != 0)
    {
        str = con.getOpenMessage(ret);
        printf("Open failed: %s\n", str.c_str());
        Aria::shutdown();
        return 1;
    }
    // set the device connection on the robot
    robot->setDeviceConnection(&con);
    // try to connect, if we fail exit
    if (!robot->blockingConnect())
    {
        printf("Could not connect to robot... exiting\n");
        Aria::shutdown();
        return 1;
    }
    // user task
```

```
    robot->addUserTask("runrobot", 50, &runrobotCB);
    // turn on the motors, turn off amigobot sounds
    robot->comInt(ArCommands::SONAR, 1);
    robot->comInt(ArCommands::ENABLE, 1);
    robot->comInt(ArCommands::SOUNDTOG, 0);
    // start the robot running, true so that if we lose connection the run stops
    //gripper->
 // setup colsole properties
 hdin=GetStdHandle(STD_INPUT_HANDLE);
 hdout=GetStdHandle(STD_OUTPUT_HANDLE);
 xycoord.X=CONX;
 xycoord.Y=CONY;
 SetConsoleMode(hdin,ENABLE_LINE_INPUT|ENABLE_ECHO_INPUT);
 SetConsoleScreenBufferSize(hdout,xycoord);
 SetConsoleTextAttribute(hdout,FOREGROUND_WHITE|BACKGROUND_BLUE);
 clrscr();
 logo();
 mmenu();
 mainmenu:
 switch(choice)
     {
        case 49:
            {
            gotoxy(0,0);
            run_sequence();
            logo();mmenu();break;
            }
        case 50:
            {
            logo();mmenu();break;
            }
        case 51:
            {
            logo();mmenu();break;
            }
        case 52:
            {
            logo();mmenu();break;
            }
        case 27: goto quitprogram;
     }
 get_option();
 goto mainmenu;
 quitprogram:
 // restorrig board and console before quiting
 xycoord.X=0;
 xycoord.Y=0;
 FillConsoleOutputAttribute(hdout,FOREGROUND_GREEN|FOREGROUND_RED|FOREGROUND_BLUE,
    2000,xycoord,&bw);
 FillConsoleOutputCharacter(hdout,fillchar,2000,xycoord,&bw);
 FillConsoleOutputAttribute(hdout,FOREGROUND_GREEN|FOREGROUND_RED|FOREGROUND_BLUE,
    2000,xycoord,&bw);
 SetConsoleTextAttribute(hdout,FOREGROUND_GREEN|FOREGROUND_BLUE|FOREGROUND_RED);
 gotoxy(26,12);
}
//###########################################################################
//###########################################################################
```

```
//#########################################################################
//______________________//_____________________//_____________________
//                       Function Definitions
//______________________//_____________________//_____________________
_inline void get_option(void)
{
 gotoxy(50,15);
 _cputs("?");
 gotoxy(50,15);
 choice='z';
 while( ((choice<49) || (choice>57)) & (choice!=27)) choice=_getch();
 gotoxy(50,15);
 cprintf("%c",choice);
}
//______________________//_____________________//_____________________
_inline void gotoxy(short a, short b) {
    xycoord.X=a;
    xycoord.Y=b;
    SetConsoleCursorPosition(hdout,xycoord);
}
//______________________//_____________________//_____________________
_inline void clrscr(void) {   xycoord.X=0;
    xycoord.Y=0;
    FillConsoleOutputAttribute(hdout,FOREGROUND_WHITE|BACKGROUND_BLUE,2000,
        xycoord,&bw);
    FillConsoleOutputCharacter(hdout,fillchar,2000,xycoord,&bw);
}
//______________________//_____________________//_____________________
_inline void mmenu(void)
{
  gotoxy(0,8);
  _cputs("                    ÉÍÍÍÍÍÍÍÍÍÍÍ MAIN MENU ÍÍÍÍÍÍÍÍÍÍÍ» ");
  gotoxy(0,9);
  _cputs("                   É¼                                  È»");
  gotoxy(0,10);
  _cputs("                   º    1 - Run Experiment             º");
   gotoxy(0,11);
  _cputs("                   º                                   º");
  gotoxy(0,12);
  _cputs("                   º    *************************      º");
  gotoxy(0,13);
  _cputs("                   º    ** SET PROCESS PRIORITY **     º");
  gotoxy(0,14);
  _cputs("                   º                                   º");
  gotoxy(0,15);
  _cputs("                   È»    Choice (or Esc to quit)      É¼");
  gotoxy(0,16);
  _cputs("                    ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
}
//______________________//_____________________//_____________________
_inline void logo(void) {
  SetConsoleTextAttribute(hdout,FOREGROUND_WHITE|BACKGROUND_BLUE);
  gotoxy(0,2);
  _cputs("                               BCI - Wrist Experiment 01");
```

```
  gotoxy(0,4);
  _cputs("                                  April 14, 2004");
 SetConsoleTextAttribute(hdout,FOREGROUND_WHITE|BACKGROUND_BLUE);
}
//_____________________//_____________________//____________________
void browse_any_file(void) {
 char f_to_run[40]="browse.com ";
 gotoxy(67,19);cputs("°");
 gotoxy(56,20);
 cputs("ÉÍÍÍÍÍÍÍÍÍÍÍÊÍÍÍÍÍÍÍÍÍÍÍ»");gotoxy(56,21);
 cputs("°   Enter File Name :  °");gotoxy(56,22);
 cputs("°                      °");gotoxy(56,23);
 cputs("ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼");
 gotoxy(65,22);
 _cscanf("%s",&filename);
 strcat(f_to_run,filename);
 gotoxy(56,20);
 cputs("                       ");gotoxy(56,21);
 cputs("                       ");gotoxy(56,22);
 cputs("                       ");gotoxy(56,23);
 cputs("                       ");
 gotoxy(61,20);cputs("   Back");gotoxy(61,20);
 system(f_to_run);
 gotoxy(67,19);cputs(" ");
 Sleep(500);
 gotoxy(61,20);
 cputs("                        ");
}
```

## C.2 eegconv.m

```
% This file was obtained from the BCI group
% and was not developed by Hinrik J. Atlason.
function [s] = EEGConv(filename)
f = fopen(filename);
s = fread(f, [1 inf], 'uint8');
fclose(f);
for i = (size(s,2)/2):-1:1
    ss(i) = s(2*i-1)*256 + s(2*i);
end
sss = reshape(ss, [24 size(ss, 2)/24]);
sss = sss - 28672;
sss = sss / 363.63;
s = sss;
```

## C.3 preprocess.m

```
% This file combines and splits the data where appropriate.
for xxxx=1:3
    if xxxx==1,
        s = 'p';
```

```
elseif xxxx==2
    s = 'j';
elseif xxxx==3
    s = 'e';
end
for x=1:2
    if x==1
        c = 'm';
    else
        c = 's';
    end
    RightFull = [];
    LeftFull = [];
    ForwardFull = [];
    BackwardFull = [];
    for xx=1:2 % 1 or 2
        if xx==1
            n = '1';
        else
            n = '2';
        end
        for xxx=1:2 % a or b
            if xxx==1
                p = 'a';
            else
                p = 'b';
            end
            Data = eegconv(['raw\' n p c s]);
            f = fopen(['raw\' n p c 'm' s]);
            Markers = fscanf(f,'%d');
            fclose(f);
            i = 1;
            Markers = Temp;
            % (Right arm=1, Left arm=2, Feet=3, Mouth=4)
            SequenceA = [1,2,1,3,4,1,2,4,3,3,4,1,2,4,3,3,4,2,1,2];
            SequenceB = [1,3,4,1,2,4,3,3,4,2,1,2,1,3,4,1,2,4,3,2];
            if xxx==1
                Sequence = SequenceA;
            else
                Sequence = SequenceB;
            end
            seq=1;
            Right=[];
            Left=[];
            Forward=[];
            Backward=[];
            for i=1:7:size(Markers,2)
                if( Sequence(seq)==1 )
                    for j=0:5
                        start = size(Right,2)+1;
                        stop = start+(Markers(i+j+1)-Markers(i+j));
                        Right(1:24,start:stop)=Data(1:24,Markers(i+j):
                            Markers(i+j+1));
                        Right(25,start)=1;
                        if (i+j+1)==(i+6)
                            Right(25,stop)=1;
                        end
                    end
                end
                if( Sequence(seq)==2 )
```

```
                    for j=0:5
                        start = size(Left,2)+1;
                        stop = start+(Markers(i+j+1)-Markers(i+j));
                        Left(1:24,start:stop)=Data(1:24,Markers(i+j):
                            Markers(i+j+1));
                        Left(25,start)=1;
                        if (i+j+1)==(i+6)
                            Left(25,stop)=1;
                        end
                    end
                end
                if( Sequence(seq)==3 )
                    for j=0:5
                        start = size(Forward,2)+1;
                        stop = start+(Markers(i+j+1)-Markers(i+j));
                        Forward(1:24,start:stop)=Data(1:24,Markers(i+j):
                            Markers(i+j+1));
                        Forward(25,start)=1;
                        if (i+j+1)==(i+6)
                            Forward(25,stop)=1;
                        end
                    end
                end
                if( Sequence(seq)==4 )
                    for j=0:5
                        start = size(Backward,2)+1;
                        stop = start+(Markers(i+j+1)-Markers(i+j));
                        Backward(1:24,start:stop)=Data(1:24,Markers(i+j):
                            Markers(i+j+1));
                        Backward(25,start)=1;
                        if (i+j+1)==(i+6)
                            Backward(25,stop)=1;
                        end
                    end
                end
                seq = seq+1;
            end % Finnished with markers.
            RightFull = [RightFull Right];
            LeftFull = [LeftFull Left];
            ForwardFull = [ForwardFull Forward];
            BackwardFull = [BackwardFull Backward];
            % a or b
        end
        % 1 or 2
    end
    f = fopen(['preprocessed\right' c s], 'wb');
    fwrite(f, RightFull, 'double');
    fclose(f);
    f = fopen(['preprocessed\left' c s], 'wb');
    fwrite(f, LeftFull, 'double');
    fclose(f);
    f = fopen(['preprocessed\forward' c s], 'wb');
    fwrite(f, ForwardFull, 'double');
    fclose(f);
    f = fopen(['preprocessed\backward' c s], 'wb');
    fwrite(f, BackwardFull, 'double');
    fclose(f);
end % m or s
```

```
end % 1 or 2 (subject)
```

## C.4 bandpass.m

```
function Output = bandpass(Input,n,low,high,channel1,channel2)
Wn=[low high]/128;
for i=channel1:channel2
    [b,a]=butter(n,Wn);
    if i==channel1
        Output=filtfilt(b,a,Input(i,:));
    else
        Output=[Output; filtfilt(b,a,Input(i,:))];
    end
end
```

## C.5 extractsegment.m

```
function [Output, last] = extractSegment(Input,first,segment)
Output = []; count = 0;
if segment == 0
    from = 1;
    to = 7;
elseif segment == 0.5
    from = 1;
    to = 2;
else
    from = segment;
    to = segment+1;
end
keeprunning = 1;
for i=first:size(Input,2)
    if keeprunning == 1
        if Input(25,i) == 1
            count = count+1;
            if count == from
                m1 = i;
            elseif count == to
                m2 = i;
                Output = Input(:,m1:m2);
            end
            if count == 7
                keeprunning = 0;
                last = i;
            end
        end
    end
end
if segment == 0.5
    Temp = Output(:,1:128);
    Output = [];
    Output = Temp;
end
```

## C.6 psd.m

```
function Output = psd(Input)
Temp = dspdata.psd(periodogram(Input));
Output = Temp.data;
```

## C.7 calculateenergy.m

```
function Output = calculateEnergy(Input)
Output = []; TempEnergy = []; e = 0;
length = size(Input,2);
for y = 1:length
    e = e + abs(Input(:,y)).^2;
end
Output = e/length;
```

## C.8 perceptron.m

```
clear all;
SAMPLENUMBER = 1; FILTERORDER = 4; FILTERLOWER = 0.5; FILTERHIGHER
= 45;
experiment = 's';
f = fopen(['preprocessed/' 'right' experiment 'p']);
RawDataRightS1 = fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'left' experiment 'p']); RawDataLeftS1 =
fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'right' experiment 'e']); RawDataRightS2 =
fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'left' experiment 'e']); RawDataLeftS2 =
fread(f, [25 inf], 'double'); fclose(f);
Accuracy = []; Mse = [];
for run = 1:3
    if run == 1
        tr1=1;tr2=7;
        te1=8;te2=13;
    elseif run == 2
        tr1=8;tr2=13;
        te1=14;te2=20;
    else
        te1=1;te2=7;
        tr1=14;tr2=20;
    end
    s1ri = 0; s2ri = 0; s1li = 0; s2li = 0;
    Train = [];
    Target = [];
    Val.P = [];
    Val.T = [];
    Test = [];
    TestTarget = [];
    for i = 1:20
        [S1rs, s1ri] = extractSegment(RawDataRightS1, s1ri + 1, SAMPLENUMBER);
        [S2rs, s2ri] = extractSegment(RawDataRightS2, s2ri + 1, SAMPLENUMBER);
        [S1ls, s1li] = extractSegment(RawDataLeftS1, s1li + 1, SAMPLENUMBER);
```

```
[S2ls, s2li] = extractSegment(RawDataLeftS2, s2li + 1, SAMPLENUMBER);
TrainTempRightS1 = [];
TrainTempLeftS1 = [];
TrainTempRightS2 = [];
TrainTempLeftS2 = [];
TestTempRightS1 = [];
TestTempLeftS1 = [];
TestTempRightS2 = [];
TestTempLeftS2 = [];
ValTempRightS1 = [];
ValTempLeftS1 = [];
ValTempRightS2 = [];
ValTempLeftS2 = [];
for c = 1:5
    if c == 1
        FROMCHANNEL = 1;
        TOCHANNEL = 1;
    elseif c == 2
        FROMCHANNEL = 2;
        TOCHANNEL = 2;
    elseif c == 3
        FROMCHANNEL = 5;
        TOCHANNEL = 5;
    elseif c == 4
        FROMCHANNEL = 6;
        TOCHANNEL = 6;
    elseif c == 5
        FROMCHANNEL = 20;
        TOCHANNEL = 20;
    end
    S1rf = bandpass(S1rs, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
        TOCHANNEL);
    S2rf = bandpass(S2rs, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
        TOCHANNEL);
    S1lf = bandpass(S1ls, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
        TOCHANNEL);
    S2lf = bandpass(S2ls, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
        TOCHANNEL);
    S1rv = mean(S1rf);
    S2rv = mean(S2rf);
    S1lv = mean(S1lf);
    S2lv = mean(S2lf);
    if tr1 <= i && i < tr2
        TrainTempRightS1 = [TrainTempRightS1; S1rv];
        TrainTempLeftS1 = [TrainTempLeftS1; S1lv];
        TrainTempRightS2 = [TrainTempRightS2; S2rv];
        TrainTempLeftS2 = [TrainTempLeftS2; S2lv];
        if size(TrainTempRightS1,1) == 5
            Target = [Target [1;-1] [-1;1] [1;-1] [-1;1]];
        end
    elseif te1 <= i && i < te2
        TestTempRightS1 = [TestTempRightS1; S1rv];
        TestTempLeftS1 = [TestTempLeftS1; S1lv];
        TestTempRightS2 = [TestTempRightS2; S2rv];
        TestTempLeftS2 = [TestTempLeftS2; S2lv];
        if size(TestTempRightS1,1) == 5
            TestTarget = [TestTarget [1;-1] [-1;1] [1;-1] [-1;1]];
        end
```

```
            else
                ValTempRightS1 = [ValTempRightS1; S1rv];
                ValTempLeftS1 = [ValTempLeftS1; S1lv];
                ValTempRightS2 = [ValTempRightS2; S2rv];
                ValTempLeftS2 = [ValTempLeftS2; S2lv];
                if size(ValTempRightS1,1) == 5
                    Val.T = [Val.T [1;-1] [-1;1] [1;-1] [-1;1]];
                end
            end
        end
        Train = [Train TrainTempRightS1 TrainTempLeftS1 TrainTempRightS2
            TrainTempLeftS2];
        Test = [Test TestTempRightS1 TestTempLeftS1 TestTempRightS2 TestTempLeftS2];
        Val.P = [Val.P ValTempRightS1 ValTempLeftS1 ValTempRightS2 ValTempLeftS2];
    end
    net = newff(minmax(Train),[2],{'purelin'},'traingd');
    net.trainParam.lr = 0.05;
    net.trainParam.epochs = 500;
    net.performFcn = 'msereg';
    net.trainParam.goal = 1e-5;
    net = init(net);
    net = train(net, Train, Target, [], [], Val);
    Y = sim(net, Test);
    correctright = 0;
    correctleft = 0;
    for i = 1:size(Y,2)
        if mod(i,2)==1 && Y(1,i)>Y(2,i)
            correctright = correctright + 1;
        elseif mod(i,2)==0 && Y(1,i)<Y(2,i)
            correctleft = correctleft + 1;
        end
    end
    Accuracy = [Accuracy; [(correctright+correctleft)/size(Y,2)*100 correctright
        correctleft] ];
    Mse = [Mse mse(TestTarget-Y)];
end
mean(Accuracy) mean(Mse)
```

## C.9 rbf.m

```
clear all;
SAMPLENUMBER = 1; FILTERORDER = 4; FILTERLOWER = 0.5; FILTERHIGHER
= 45;
experiment = 's';
f = fopen(['preprocessed/' 'right' experiment 'p']);
RawDataRightS1 = fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'left' experiment 'p']); RawDataLeftS1 =
fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'right' experiment 'e']); RawDataRightS2 =
fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'left' experiment 'e']); RawDataLeftS2 =
fread(f, [25 inf], 'double'); fclose(f);
Accuracy = []; Mse = [];
for run = 1:5
    splitt = run*4;
```

```
s1ri = 0; s2ri = 0; s1li = 0; s2li = 0;
Train = [];
Target = [];
Val.P = [];
Val.T = [];
Test = [];
TestTarget = [];
for i = 1:20
    [S1rs, s1ri] = extractSegment(RawDataRightS1, s1ri + 1, SAMPLENUMBER);
    [S2rs, s2ri] = extractSegment(RawDataRightS2, s2ri + 1, SAMPLENUMBER);
    [S1ls, s1li] = extractSegment(RawDataLeftS1, s1li + 1, SAMPLENUMBER);
    [S2ls, s2li] = extractSegment(RawDataLeftS2, s2li + 1, SAMPLENUMBER);
    TrainTempRightS1 = [];
    TrainTempLeftS1 = [];
    TrainTempRightS2 = [];
    TrainTempLeftS2 = [];
    TestTempRightS1 = [];
    TestTempLeftS1 = [];
    TestTempRightS2 = [];
    TestTempLeftS2 = [];
    ValTempRightS1 = [];
    ValTempLeftS1 = [];
    ValTempRightS2 = [];
    ValTempLeftS2 = [];
    for c = 1:5
        if c == 1
            FROMCHANNEL = 1;
            TOCHANNEL = 1;
        elseif c == 2
            FROMCHANNEL = 2;
            TOCHANNEL = 2;
        elseif c == 3
            FROMCHANNEL = 5;
            TOCHANNEL = 5;
        elseif c == 4
            FROMCHANNEL = 6;
            TOCHANNEL = 6;
        elseif c == 5
            FROMCHANNEL = 20;
            TOCHANNEL = 20;
        end
        S1rf = bandpass(S1rs, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
            TOCHANNEL);
        S2rf = bandpass(S2rs, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
            TOCHANNEL);
        S1lf = bandpass(S1ls, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
            TOCHANNEL);
        S2lf = bandpass(S2ls, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
            TOCHANNEL);
        S1rv = mean(S1rf);
        S2rv = mean(S2rf);
        S1lv = mean(S1lf);
        S2lv = mean(S2lf);
        if (i <= splitt-4) || (splitt < i)
            TrainTempRightS1 = [TrainTempRightS1; S1rv];
            TrainTempLeftS1 = [TrainTempLeftS1; S1lv];
            TrainTempRightS2 = [TrainTempRightS2; S2rv];
            TrainTempLeftS2 = [TrainTempLeftS2; S2lv];
```

```
                if size(TrainTempRightS1,1) == 5
                    Target = [Target [1;-1] [-1;1] [1;-1] [-1;1]];
                end
            else
                TestTempRightS1 = [TestTempRightS1; S1rv];
                TestTempLeftS1 = [TestTempLeftS1; S1lv];
                TestTempRightS2 = [TestTempRightS2; S2rv];
                TestTempLeftS2 = [TestTempLeftS2; S2lv];
                if size(TestTempRightS1,1) == 5
                    TestTarget = [TestTarget [1;-1] [-1;1] [1;-1] [-1;1]];
                end
            end
        end
        Train = [Train TrainTempRightS1 TrainTempLeftS1 TrainTempRightS2
            TrainTempLeftS2];
        Test = [Test TestTempRightS1 TestTempLeftS1 TestTempRightS2 TestTempLeftS2];
    end
    net = newrb(Train,Target,0.05);
    Y = sim(net, Test);
    correctright = 0;
    correctleft = 0;
    for i = 1:size(Y,2)
        if mod(i,2)==1 && Y(1,i)>Y(2,i)
            correctright = correctright + 1;
        elseif mod(i,2)==0 && Y(1,i)<Y(2,i)
            correctleft = correctleft + 1;
        end
    end
    Accuracy = [Accuracy; [(correctright+correctleft)/size(Y,2)*100 correctright
        correctleft] ];
    Mse = [Mse mse(TestTarget-Y)];
end
mean(Accuracy) mean(Mse)
```

## C.10 lvq.m

```
clear all;
SAMPLENUMBER = 1; FILTERORDER = 4; FILTERLOWER = 0.5; FILTERHIGHER
= 45;
experiment = 's';
f = fopen(['preprocessed/' 'right' experiment 'p']);
RawDataRightS1 = fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'left' experiment 'p']); RawDataLeftS1 =
fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'right' experiment 'e']); RawDataRightS2 =
fread(f, [25 inf], 'double'); fclose(f); f =
fopen(['preprocessed/' 'left' experiment 'e']); RawDataLeftS2 =
fread(f, [25 inf], 'double'); fclose(f);
Accuracy = []; Mse = [];
for run = 1:5
    splitt = run*4;
    s1ri = 0; s2ri = 0; s1li = 0; s2li = 0;
    Train = [];
    Target = [];
```

```
Val.P = [];
Val.T = [];
Test = [];
TestTarget = [];
for i = 1:20
    [S1rs, s1ri] = extractSegment(RawDataRightS1, s1ri + 1, SAMPLENUMBER);
    [S2rs, s2ri] = extractSegment(RawDataRightS2, s2ri + 1, SAMPLENUMBER);
    [S1ls, s1li] = extractSegment(RawDataLeftS1, s1li + 1, SAMPLENUMBER);
    [S2ls, s2li] = extractSegment(RawDataLeftS2, s2li + 1, SAMPLENUMBER);
    TrainTempRightS1 = [];
    TrainTempLeftS1 = [];
    TrainTempRightS2 = [];
    TrainTempLeftS2 = [];
    TestTempRightS1 = [];
    TestTempLeftS1 = [];
    TestTempRightS2 = [];
    TestTempLeftS2 = [];
    ValTempRightS1 = [];
    ValTempLeftS1 = [];
    ValTempRightS2 = [];
    ValTempLeftS2 = [];
    for c = 1:5
        if c == 1
            FROMCHANNEL = 1;
            TOCHANNEL = 1;
        elseif c == 2
            FROMCHANNEL = 2;
            TOCHANNEL = 2;
        elseif c == 3
            FROMCHANNEL = 5;
            TOCHANNEL = 5;
        elseif c == 4
            FROMCHANNEL = 6;
            TOCHANNEL = 6;
        elseif c == 5
            FROMCHANNEL = 20;
            TOCHANNEL = 20;
        end
        S1rf = bandpass(S1rs, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
            TOCHANNEL);
        S2rf = bandpass(S2rs, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
            TOCHANNEL);
        S1lf = bandpass(S1ls, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
            TOCHANNEL);
        S2lf = bandpass(S2ls, FILTERORDER, FILTERLOWER, FILTERHIGHER, FROMCHANNEL,
            TOCHANNEL);
        S1rv = mean(S1rf);
        S2rv = mean(S2rf);
        S1lv = mean(S1lf);
        S2lv = mean(S2lf);
        if i < 14
            TrainTempRightS1 = [TrainTempRightS1; S1rv];
            TrainTempLeftS1 = [TrainTempLeftS1; S1lv];
            TrainTempRightS2 = [TrainTempRightS2; S2rv];
            TrainTempLeftS2 = [TrainTempLeftS2; S2lv];
            if size(TrainTempRightS1,1) == 5
                Target = [Target 1 2 1 2];
            end
```

```
            else
                TestTempRightS1 = [TestTempRightS1; S1rv];
                TestTempLeftS1 = [TestTempLeftS1; S1lv];
                TestTempRightS2 = [TestTempRightS2; S2rv];
                TestTempLeftS2 = [TestTempLeftS2; S2lv];
                if size(TestTempRightS1,1) == 5
                    TestTarget = [TestTarget 1 2 1 2];
                end
            end
        end
        Train = [Train TrainTempRightS1 TrainTempLeftS1 TrainTempRightS2
            TrainTempLeftS2];
        Test = [Test TestTempRightS1 TestTempLeftS1 TestTempRightS2 TestTempLeftS2];
    end
    net = newlvq(minmax(Train),18,[0.5 0.5], 0.05);
    net.trainParam.epochs = 500;
    net.trainParam.goal = 1e-5;
    net = train(net, Train, ind2vec(Target));
    Y = sim(net, Test);
    Y = vec2ind(Y);
    correctright = 0;
    correctleft = 0;
    for i = 1:2:size(Y,2)
        if Y(i) == 1, correctright = correctright + 1;
        elseif Y(i+1) == 2, correctleft = correctleft + 1;
        end
    end
    Accuracy = [Accuracy; [(correctright+correctleft)/size(Y,2)*100 correctright
        correctleft] ];
    Mse = [Mse mse(TestTarget-Y)];
end
mean(Accuracy) mean(Mse)
```

# Appendix D

# Project management

*Figures and diagrams that relate to project management are given here.*

| | **Tasks** | **Milestones** |
|---|---|---|
| $Week1$(20/6 – 26/6) | Literature gathering & reading: *A number of books were obtained from the library on neuroscience, neural networks, feature extraction and filtering. Papers include Essex university research papers on BCI systems and neural networks.* | |
| $Week2$(27/6 – 3/7) | Literature gathering & reading: *Further reading of practical applications of BCI systems, neuroscience and feature extraction. Techniques for data filtering and training data pre processing were also considered. A procedure for training data gathering was formulated and is pending review. The next step will most likely be to schedule a session where training data is gathered (brain wave signals). Further literature gathering & reading will proceed throughout the next weeks as necessary.* | Milestone 1: reached on time. |
| $Week3$(4/7 – 10/7) | Analysis & documentation: *Analysis continues for the feature selection part. Relevant techniques are documented for later use.* | |
| $Week4$(11/7 – 17/7) | Analysis & documentation: *Analysis continues for the feature extraction part. Relevant techniques are documented for later use. Some revision of the project plan is needed since background research took longer time than expected and implementation started before scheduled.* | Milestone 2: Not reached. |
| $Week5$(18/7 – 24/7) | System design & documentation: *Design begins. Implementation proceeds and the code for the experiment is completed. Documentation is carried out in parallel to the design. The finished parts of the report are now as follows: layout and structure, feature extraction references and keywords, some considerations regarding the implementation.* | |

| | **Tasks** | **Milestones** |
|---|---|---|
| *Week*6(25/7 – 31/7) | System design & documentation + Implementation + Testing & result gathering: *Implementation proceeds. Basic filtering is done on borrowed data. The borrowed data is used to explore different techniques and MATLABs implementations of popular methods. The experiments require the robots to have lights which serve as clues in the experiment. This should be ready next week.* | Milestone 3: Reached on time. |
| *Week*7(1/8−7/8) | Implementation + Testing & result gathering: *Code for experiments ready. Pioneer robots in the labs been fitted with equipment which omits light clues during the experiment. Implementation of signal processing part proceeds.* | Milestone 4: Reached on time. |
| *Week*8(8/8 – 14/8) | Testing & result gathering + Implementation & Testing documented: *Implementation, Testing and result gathering continue. Experiment 1 completed and that data is used for testing. The results can not be compared to data from subject 2 since it was too contaminated with muscle signals. Documentation of testing falls behind because of corrupt data from experiment 2. Hence, experiment 3 is scheduled for next week.* | Milestone 5: Partly reached. |
| *Week*9(15/8 – 21/8) | Implementation & Testing documented: *Continuing with tasks from last week. There have been some delays because of the corrupt data. Also, implementation took longer than expected.* | Milestone 6: Not reached. |
| *Week*10(22/8 – 28/8) | Revision of results and documents: *Implementation and Testing documented. Additional analysis of data. Preparation for presentation begins.* | Milestone 7: Reached on time |
| *Week*11(29/8 – 4/9) | Preparation for presentation: *Preparation for presentation continues. Presentation scheduled on Friday the 2nd.* | |
| *Week*12(5/9 – 11/9) | Documentation | |
| *Week*13(12/9 – 18/9) | Documentation | |

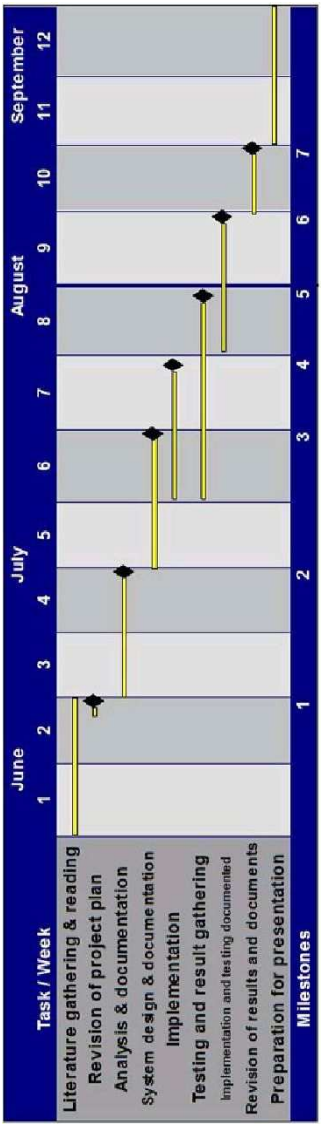



Figure D.1: The original Gantt diagram.