Implementation of Online Brain Computer Interface Using Motor Imagery In LabVIEW.

Anurag Garg
Asbjørn Binderup

Semester 9-10

Date: 7th June 2007

Aalborg University
Department of Health Science And Technology

Department of Health Science and Technology, Aalborg University

TITLE:

Implementation of Online Brain Computer Interface Using Motor Imagery In LabVIEW

PROJECT PERIOD:

1st September, 2006 - 7th June, 2007

SEMESTER:

9th-10th

GROUP:

Group 1085a

GROUP MEMBERS:

Anurag Garg
Asbjørn Binderup

SUPERVISORS:

Alvaro Fuentes Cabrera
Kim Dremstrup Nielsen

NUMBER OF COPIES: 5

NUMBER OF PAGES: 72

APPENDIX: 33

ABSTRACT

The purpose of this study was to demonstrate the effect of training on a subject's performance using an online Brain Computer Interface system with feedback.
In order to do so, we developed an Online system BCILab which performs data acquisition, signal processing, feature extraction, classification and saving of all parameters. The data acquisition was done using a TCP connection with the ACQUIRE program. Spatial filter (common average reference) and detrending were used during signal processing to achieve high signal to noise ratio and stationarity of signal respectively. The power spectrum calculated for the autoregressive coefficients for channel C3 and C4 at 8, 10 and 12 Hz were used as the features for classification by a bayesian classifier.
The data for training of the classifier was collected during a single screening session and multiple feedback sessions when the subject was asked to perform the left or right motor imagery task depending on target state. The performance of the subject was analysed over each session and over days.
As the study had only two subjects, it was considered as a case study. However, in case of both subjects the performance of over days has shown improvement, in case of subject 1 it improved from (48%±15%) to (67%±10%) and for subject 2 (50%±0%) to (56%±11%) over 3 and 4 days respectively.

Keywords:
Electroencephalogram,
Mu Rhythms,
Motor Imagery,
Autoregressive Modelling,
Bayesian Classifier.

Acknowledgement

We would like to thank our supervisors, Alvaro Fuentes Cabrera and Kim Dremstrup Nielsen, for their guidance, ideas and patience. We would also like to thank Omar Feix Do Nascimento for his guidance in making connectors and setup files for the system. Clemens Eder gets our thanks for providing the program managing the TCP connection with ACQUIRE. Finally, we would like to thank Department of Health Science and Technology, Aalborg University for providing us all the necessary needs for the project.

Aalborg University, June 2007
BME GROUP 1085a

Anurag Garg

Asbjørn Binderup

Abbreviations

AAR	Adaptived Autoregression
AR	Autoregression
AIC	Akaike Information Criterion
BCI	Brain Computer Interface
CAR	Common Average Reference
EEG	Electroencephalogram
EKG	Electrocardiogram
EMG	Electromyogram
ERD	Event Related Desynchronisation
ERP	Event Related Potentials
ERS	Event Related Synchronisation
FFT	Fast Fourier Transform
FLD	Fisher's Linear Discrimminant
FPE	Function Prediction Error
fMRI	Functional Magnetic Resonance Imaging
IIR	Infinite Impulse Response
MEM	Max Entropy Method
MI	Motor Imiginary
NMI	Non-Motor Imiginary
WSS	Wide Sense Stationary
SL_FDM	Surface Laplacian using Finite Difference Methods
SL_SS	Surface Laplacian using Spherical Spline
SMA	Supplementary Motor Area
SNR	Signal to Noise Ratio
TCP	Transmission Control Protocol

CONTENTS

CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

When we talk about interfacing with a computer, we usually talk about usage of a keyboard or a mouse which in other words means controlled through physical activity. The idea of our project is to make an interface with a computer, but using changes in brain activity rather than a real physical movement as controller. In order to get insight into brain activity we analyze Electroencephalogram (EEG). EEG is electrical recordings from the scalp that are produced by the neurons present in the outer layer of the brain. These raw EEG signals can be converted into control signals using signal processing techniques and statistical methods mentioned in later sections. This way of contact between the brain to the external environment as to control a device using a computer interface is commonly known as Brain Computer Interface (BCI). Also, it can be described as a direct technological interface between a brain and a computer not requiring any physical movement from the user.

1.1 BCI History

In other words, BCI provide their users a communication channel that does not depend on the output of the brain to the peripheral nerves and muscles. The main interest of BCI lies in the rehabilitation of patients with severe motor disabilities - disabilities making them incapable of movements involving voluntary control. The early research in the field goes back to 1967 when Dewan used biofeedback training (where the subject has the feedback how well he or she is controlling the device) to modulate the occipital alpha rhythm in order to transmit Morse code messages. In 1988 Farwell and Donchin at University of Illinois used a technique for detecting P300 component of a subject's Event Related Brain Potential (ERP) and used it to select from an array of 36 screen positions. They placed an electrode over the parietal cortex and the signals were processed using a peak picking algorithm which identifies P300. The system could communicate an average of 2.3 characters per minute [5]. In 1994 McFarland and Wolpaw at the New York State Department of Health trained subjects to control their mu rhythm (alpha rhythms over motor cortex). The amplitude of mu rhythm was used to control cursor movement on computer screen. Two electrodes were placed over the motor cortex area in the sensorimotor cortex area. Fast Fourier Transform (FFT) was used to calculate the 9Hz EEG component (alpha band)in real time. The FFT was used to control graded vertical movement of the cursor. A target at a randomly chosen height and fixed width then made its

way from left to right edge of the screen in a 8 seconds period. The subject was told to move the cursor on the right edge of the screen so as to intercept the target. The best subject after a training period of a few weeks was able to hit 75 percents of the targets.

Over the past 10 years, research in this field has grown tremendously. In 1995, there were not more than 6 groups working in it but now there are more than 20 [6]. One of the major interest areas for researchers is look at the physiological signals for communication and operation of devices for severely motor disabled as well as healthy people. In particular a few groups like Wolpaw and McFarland [6], Kalcher [7] and Millan [8] are working on recognition of EEG signals during thought on different tasks. The type of task can be divided into two categories, Motor Imagery (MI) and Non Motor Imagery (NMI) where MI consists of imagining a real movement, for example when asking a subject to think of moving his hand or leg. NMI consists of imagination like asking a subject to think that he is listening to a music or he is standing in a room. Hence, the choice of imaginary task type is very important for the type of signal processing the BCI has to use to extract features for classification. Researchers such as Birbaumer [9] and his team mates measured shifts of slow cortical potential over vertex whereas Wolpaw [10] and co-workers [11] focus on sensorimotor cortex, they measure continuous changes in the amplitudes of mu and beta rhythms. Pfurtscheller in contrary has focused on the issue that the event related phenomena represents frequency specific changes of the ongoing EEG activity and may consist,in general terms,either increase or decrease of power in given frequency bands. He has attributed this could be due to increase or decrease of synchrony of the underlying neuronal populations respectively. The decrease is called as Event Related Desynchronisation (ERD) whereas increase as Event Related Synchronization (ERS) [12].

In 1992, PfurtScheller designed the online graz ***BCI I*** system. One subject was trained to use the system for over four and a half hour session. The subject was set to the task of moving a cursor on a screen either left or right depending on the target. The protocol was the following: A beep prepared the user for the start of the task, after a few seconds a cursor appeared in the middle of the screen indicating that the user should press the appropriate switch. About 80 such trials were performed during the session. Using the electrodes C3 and C4 and a neural network classification of autoregressive features the classifier predicted 70% of movement correctly. After four such sessions this increased to 85%. This points proved that more training data improves the classification result [13].

Later in 1996, PfurtScheller showed how left and right index finger movement and toe and tongue movement could be differentiated by their ERD/ERS's. They used the concept that finger or hand movement is accompanied by a blocking of mu rhythm and by a short Gamma rhythm. He used 8 electrodes in a rectangular array over the sensorimotor cortex area and calculated power spectral estimates for the following: 10-12Hz (finger and tongue), 30-33Hz (toe), 38-40Hz (finger and tongue). They were calculated for periods of 250ms. With the help of this study, he developed online graz ***BCI II*** and it could control this movements, right and left index finger and a right foot movement. Four subjects were trained to use the interface over four sessions of one and half hour each in a period of 2 weeks. Each session had four blocks of 60 trials with 5 minute break in between. In each trial the subject was seated looking at a cross on a computer screen. One second after a beep, an arrow appeared

pointing left, right or down. When the arrow disappeared after 1.25s, the subject pressed a switch with his right or left index finger or moved the toes upwards. The EEG signals in the second before movement were classified by a learning vector quantifier. Data from the first session were used to train the classifiers. The system was able to correctly classify the data with 80% of success [14].

However, the main problem with the BCI systems developed by 2000 was that they were designed for a specific BCI method and therefore not suited for the systematic studies that are essential for continued progress [15]. Addressing this problem, a group of researchers Gerwin Schalk, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R.Wolpaw developed a general purpose research and development platform ***BCI2000*** at the Wadsworth Center of the New York State Department of Health in Albany, New York, USA. BCI2000 has been successful in meeting the requirements for the real time BCI Systems and has substantially reduced the labour and cost. BCI2000 can be described a system of four modules : Operator, Source, Signal Processing and Application. The source module digitizes and stores brain signals and passes them on without any further preprocessing to signal processing. It consists of a data acquisition and a data storage component. The signal processing module converts signals from the brain into signals that control an output device. This conversion has two stages: feature extraction and feature translation. And the user application module receives control signals from signal processing and uses them to drive an application. It is programmed in C++ language and uses Borland C++ as the builder. It was designed for Windows 2000/XP as a operating system [15]. For more information please visit [16].

To support BCI 2000, Alvaro Fuentes Cabrera at SMI, the Department of Health Science and Technology, Aalborg University is working (since 2004) on developing an offline analysis BCI tool ***Smario***. This system is programmed in Matlab. For further enquires about the Smario please contact Alvaro Fuentes Cabrera [17] ,Aalborg University, Denmark.

1.2 A BCI System

As described above, BCI acts a means of communication for the subjects suffering from neurological disorders that hinders the normal communication due to the loss of motor functions. We will be using a EEG based BCI system, which will use mental activity to control the external devices. The basic concept is, mental activities or imaginations of any movements produce some electrophysiological changes which can be detected by a BCI system and can be converted into a control signals to use external device. Hence a BCI system can be seen as doing three basic functions : acquiring an input signals, translating this input signals into a control signals using a translation algorithm, controlling a external device using this control signals.

Though there already exists different kinds of BCI system we have gained interest in developing our own. Both as a programming learning experience and to get a better understanding of the different elements involved in a BCI. Based on the results and documentation from previous research in the field, we have decided to put out focus on motor imagery tasks for

our BCI, due to that field being well documented and tested. As we do not plan to test the system on people with neural disabilities at this first stage of development we will aim it a "normal" people with no disabilities.

First we tried to continue developing on the NetReader program that had previously been used by Alvaro Fuentes Cabrera [17] in a study of steady-state visual evoked potentials and use this as the program for our BCI. The results of these effort can be found in appendix A. As the complexity for making this program in C++ reached beyond our skills we had to find an alternative tool.

Instead we decided to make our program in LabVIEW. LabVIEW is a visual orientated programming tool that has been in use for more than 20 years. It focus on the development of virtual instruments used for data acquisition, processing and display. The environment is aimed at real-time execution which we need for an online BCI. To record the EEG signals we will use a 64 channel EEG cap and the ACQUIRE program developed by Neuroscan for use with their amplifiers. ACQUIRE will then send the recorded signals to our program (BCILab) through TCP.

One of the task of the project is to make the BCI a modularized system, which means making it in a form so that parts or modules can be replaced with some other without changing the rest of the structure. The flow diagram for such a system is as shown in figure: 1.1

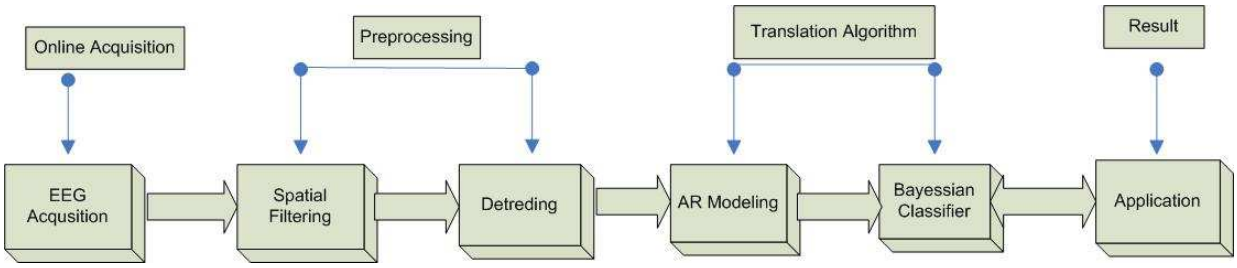

Figure 1.1: Modularized online BCI system.

An online acquisition system will acquire EEG signals from the subject. Before performing any further processing, the signals are preprocessed to make them less noisy and more significant. For this, Spatial Filtering and Detrending is performed, once the signal is processed it is then passed through an AutoRegressive (AR) filter, and using FFT a power spectrum is generated for an AR coefficients. The features extracted from the power spectrum are then classified using a Bayesian Classifier. They are classified for left or right motor imagery. Based on the classification, a visual display will indicate the the result of the subject's thought in form of left or right.

The online system also needs to save the data that is being the recorded and all paramters for the signal processing modules for any later offline analysis. We decided to try and store in the same file format as specified by BCI2000, so offline tools made for one program could work on both.

1.3 Problem Statement

In basic, a BCI system contains two adaptive components. One being the recording and classification program the other being the subject. As previously mentioned a lot of research has been done in tuning the adaptive program part through examination of filtering, feature selection and classification methods. This is often done offline with previous recorded EEG data. Instead we will focus on the training of the subject in an online system and with feedback he receives from the program.

We plan to base our selection of signal processing modules on common and proven methods within the field. After selection we do not plan to change the parameters for these modules during sessions. We will do a screening session on the untrained subject for left and right hand imaginary movement during which no classification is done to provide feedback about his performance. The screening session is used to select the best given features to classify from this session. We will use these as training data for the classifier during feedback session where the subject gets the feedback about his performance based on the classification.

We want to prove that - ***Using data from a single screening session to train a classifier, a subject can improve his classification success rate over multiple sessions when using an online program that provides feedback of his performance during the sessions***. In other words, it means that a subject performance can be improved over days of training. The classification success rate will be evaluated on the subjects ability to get the correct classification of a target imaginary movement.

Chapter 2

Neurophysiological Theory for BCI

This chapter aims at providing neurophysiological background related to BCI.

2.1 The Human Brain

The human brain can be divided into four different regions depending on their location and their function. The brief information about the different lobes is as follows, refer figure: 2.1:

Frontal Lobe

- Located in most anterior, right under the forehead.

  Functions:

- The frontal lobes have been found to play a part in impulse control, judgment, language production, working memory,motor function, problem solving,sexual behavior, socialization, and spontaneity. The frontal lobes assist in planning, coordinating, controlling, and executing behavior.

Parietal Lobe

- Located near the back and top of the head.

  Functions:

- The parietal lobe plays important roles in integrating sensory information from various parts of the body, and in the manipulation of objects. Portions of the parietal lobe are involved with visuo-spatial processing.

Occipital Lobe

- Located most posterior, at the back of the head.

  Functions:

- The function of the occipital lobe is to control vision and colour recognition.

Temporal Lobe

- Located at the side of head above the ears.

Functions:

- The temporal lobe is involved in auditory processing and also in the recognition of both speech and vision. The temporal lobe contains the hippocampus and is therefore involved in memory formation as well.

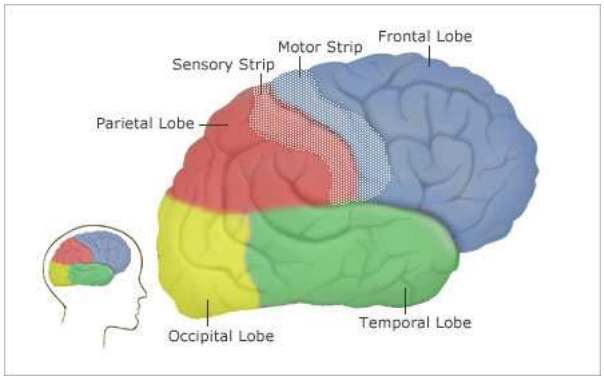

Figure 2.1: The human brain. The four lobes of the cerebrum and the motor and sensory strips.[1]

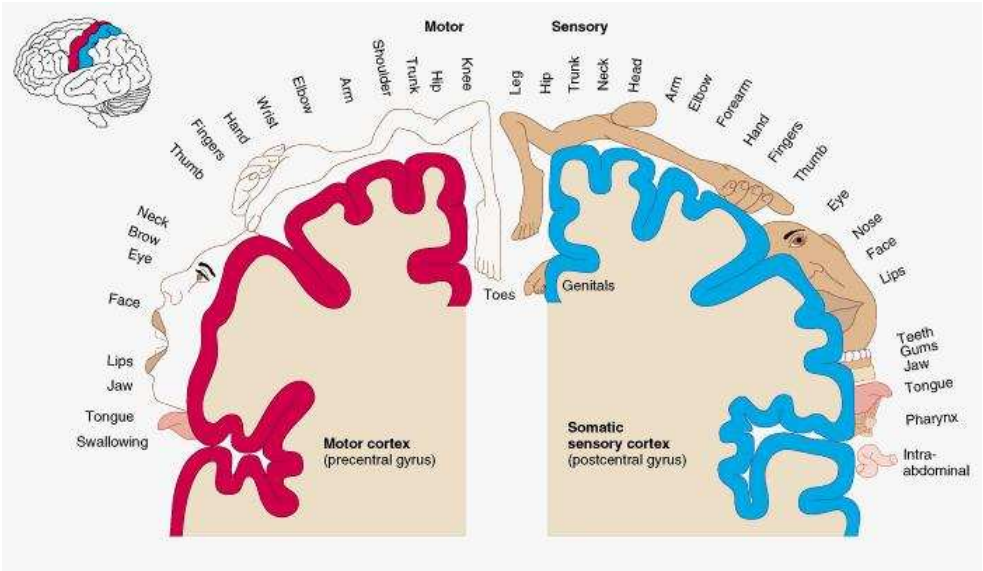

Figure 2.2: The Homunculus, representing the involvement of a body part to a corresponding region in the brain.[1]

2.2 Physiology of Movement

A simple movement from walking to moving your lips for talking involves motor function. There are many anatomical regions involved in motor functionality, the primary one being

the motor cortex. This is located in the frontal lobe and its function is to generate the impulses to control movement. Signals from primary motor cortex cross the body's midline to activate skeletal muscles on the opposite side of the body, meaning that the left hemisphere of the brain controls the right side of the body, and the right hemisphere controls the left side of the body. Every part of the body is represented in the primary motor cortex, and these representations are arranged in a particular order - the foot is next to the leg which is next to the trunk which is next to the arm and the hand. The amount of brain area devoted to any particular body part represents the amount of control that the primary motor cortex has over that body part. For example, a lot of brain space is required to control the complex movements of the hand and fingers, and these body parts have larger representations in primary motor cortex than the trunk or legs. The figure 2.2 shows this arrangement of different body parts in the motor cortex as well as the sensory cortex.

Other regions of the cortex involved in motor function are called the secondary motor cortices. These regions include the posterior parietal cortex, the premotor cortex, and the supplementary motor area (SMA). The posterior parietal cortex is involved in transforming visual information into motor commands. For example, the posterior parietal cortex would be involved in determining how to steer the arm to a glass of water based on where the glass is located in space. The posterior parietal areas send this information on to the premotor cortex and the supplementary motor area. The premotor cortex lies just in front of (anterior to) the primary motor cortex. It is involved in the sensory guidance of movement, and controls the more proximal muscles and trunk muscles of the body. The supplementary motor area lies above, or medial to, the premotor area, also in front of the primary motor cortex. It is involved in the planning of complex movements and in coordinating two-handed movements. The supplementary motor area and the premotor regions both send information to the primary motor cortex as well as to brainstem motor regions.

Information from the primary motor cortex, SMA and premotor cortex goes to the fibers of the corticospinal tract. The corticospinal tract is the only direct pathway from the cortex to the spine and is composed of over a million fibers. The corticospinal tract is the main pathway for control of voluntary movement in humans. There are other motor pathways which originate from subcortical groups of motor neurons (nuclei). These pathways control posture and balance, coarse movements of the proximal muscles, and coordinate head, neck and eye movements in response to visual targets. Subcortical pathways can modify voluntary movement through interneuronal circuits in the spine and through projections to cortical motor regions [18].

2.3 Information related to Movement in EEG

Earlier researches have shown that, during the planning or execution of any movement, certain changes occurs in EEG [19]. These changes are known as Event Related De-Synchronization and Event Related Synchronization. These information can be extracted from the recorded EEG signals. Considering these as important concepts for the project understanding along with the Motor Imagery which is the area of investigation we would like to provide brief

information on these concepts.

2.3.1 Event Related Synchronization and De synchronization

Event Related Synchronization (ERS) and Event Related De-synchronization (ERD) falls under the category of event related EEG response. They are also associated as non phase locked event unlike Event Related Potentials (ERPs)which are known as phase locked events.

ERD and ERS are measurable time limited changes in the ongoing EEG rhythm arising from specific neurons. These changes occurs in specific bands and location of cortex. Coherent activity in large neuronal pools can result in high amplitude and low frequency oscillations (eg: alpha band rhythms), whereas synchrony in localized neuronal pools can be the source of gamma oscillations [2]. The dynamic of such a network can result in phasic changes in the synchrony of cell populations due to externally or internally paced events and lead to characteristics EEG patterns. Two of such patterns are, event related desynchronization where amplitude attenuation occurs and the event related synchronization where the enhancement of specific frequency components occurs as shown in figure: 2.3 [2].

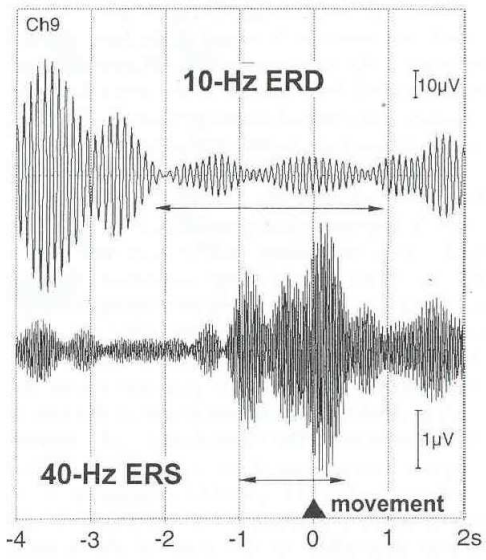

Figure 2.3: Examples of band pass filtered EEG trials from voluntary finger movement displaying desynchronization of mu rhythms and an embedded burst of gamma band oscillations. [2]

According to Pfurtscheller the study of voluntary movements can be considered as a good model to investigate the dynamics of brain oscillations. It is supposed that the cortical activity is involved in preparation,producing and controlling motor behavior, which means that motor behavior requires the activation of a large number of cortical and subcortical systems [2].

Pfurtscheller has stated that approximately 1.5-2 seconds prior to movement the mu ERD occurs, initially restricted to the contralateral hemisphere and later on, shortly before the movement-onset, also on the ipsilateral side. During the execution of the movement, ERD seems to be almost symmetric on both hemisphere and it starts recovering once the movement is offset. The gamma ERS starts approximately at the same time as that of the mu ERD but it ends before the onset of the movement. After the termination of movement, ERD recovers slowly within a few seconds in the alpha(mu) band and relatively quicker in the beta band. However, beta ERD not only shows a fast recovery but also a rebound in the form of a burst of oscillations. An illustration of ERD/ERS and movement related potentials in a simple finger movement is shown in figure: 2.4.

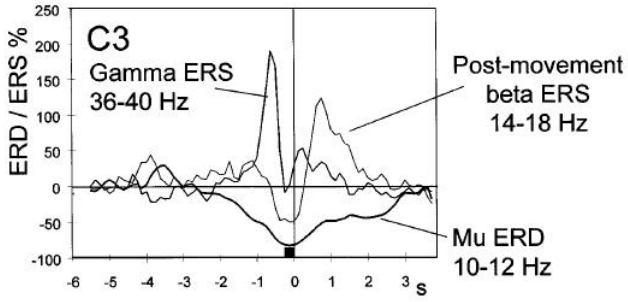

Figure 2.4: Figure showing the synchronization and desynchronization activities during a brisk finger movement. [2]

2.3.2 Motor Imagery

Motor imagery and execution of actual movement. However, Decety [20] stated that the main difference between performance and imagery is that in the latter case execution would be blocked at the cortico-spinal level. So, this fact explains that the mental rehearsal may help in motor skill learning and also in motor performance.

Hallet [21] has shown in his studies of Functional Magnetic Resonance Imaging (fMRI) there are some activation region in the primary motor cortex during motor imagery, though to a lesser extent than during the actual motor performance.

Study conducted by Pfurtscheller and Neuper [12],[2] has strongly shown that there is a activity of motor cortex in the mental simulation of movement. In the study conducted, subjects were required to perform a brisk dorsiflexion movement of the left or right hand, as directed by the arrow on the computer screen. During the imagination condition, subjects were asked to imagine the same movement but without actually making any physical real movement and being in a relaxed position.

It was observed that independent of the condition that is imagined or the real movement, the prominent changes in the EEG were localized in the hand area sensorimotor cortex. A

similar ERD has been found on the contralateral side during the motor imagery as is usually found during the planning or preparation of real movement. An example of ERD distribution during motor imagery, mapped on the reconstructed cortical surface of one subject is as shown in figure: 2.5. It can be observed from the figure that, a circumscribed ERD with a focus on the primary hand movement is clearly distinct incase of imaged movement of right or left hand movement. During an execution of a real movement, the initially contralateral ERD develops a bilateral distribution whereas in case of imagined movement it seems to be more concentrated on the contralateral side only.

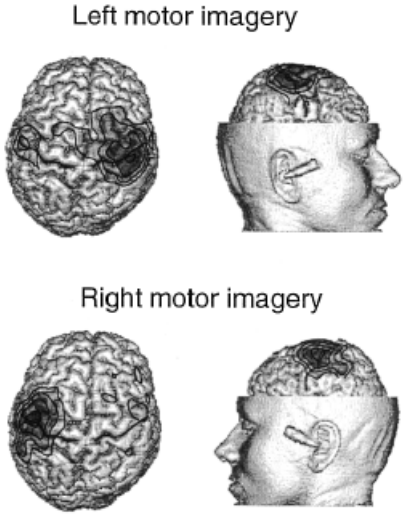

Figure 2.5: Figure showing an example of ERD maps for a single subject calculated for the cortical surface of a realistic head model for motor imagery. [2]

Imagery can be defined as the imagined execution of a real movement, means imaging a motor act without any sensory input or any output in form of muscular movements. It is believed that imaging a real movement involves similar brain regions/functions which are involved in planning.

Chapter 3

Signal Processing Tools

This chapter aims at providing the theory for the signal processing techniques used in the study. Before the signal or the data appears at the signal processing modules, it is in its raw form which means it contains a lot of noise and signals from unwanted movement. As our study is based on EEG signals from motor imagery, any signals generated from physical movement of body or eye movement are considered as artifacts. Before using the signals for actual study all this unwanted information has to be removed using proper selection of signal processing methods.

A block diagram which illustrates the signal processing tools in the project is shown in figure: 1.1.

3.1 Preprocessing

3.1.1 Spatial Filtering

The Spatial filtering of EEG data is very important when analyzing the brain activity. Spatial filtering helps in increasing signal to noise ratio (SNR) and thus provides a better classification of analyzed mental states [22] [11] [23].

As mentioned earlier, this noises could be due to Electromyogram (EMG), Electrocardiogram (EKG) and eye movement or eye blink potentials, and non-mu EEG components such as visual alpha rhythms.

However, it is very important to choose the correct spatial filtering technique. The most common techniques are Common Average Reference (CAR), Small Laplacian and Large Laplacian, Surface Laplacian using Finite Difference Methods (SL_FDM) and Surface Laplacian using Spherical Spline approach (SL_SS). The proper selection of technique is determined by the wanted control signals, like mu rhythms, and the location and strength of EEG sources and non-EEG noise. The main criteria for selection is to find a filter which provides the highest SNR. The researchers McFarland, Mourino and their co-researchers analyzed which of the above mentioned technique is best suited for EEG filtering. The research have concluded that

CAR is a better technique than the others mentioned.

McFarland supported the argument that CAR and Large Laplacian appear to be well-suited for a communication system using the mu rhythm or closely related beta activity, they would not necessarily be appropriate for systems using more broadly distributed activity, such as the P300 system described by Farwell and Donchin. For each system, the spatial filter chosen should maximally accentuate the control signal and maximally attenuate other EEG activity and non-EEG artifacts [11].

Mourino has shown in his research that SL_FDM is not efficient when dealing with fewer electrodes due to the lack of information as the filter is always under boundary restrictions. However, CAR and SL_SS has shown better results with fewer electrodes [22].

Based on the information gathered from the above mentioned researches, we decided to use CAR as a spatial filter for our study as we would be interested in analyzing the electrodes only sensor motor cortex area and motor cortex area.

CAR subtracts the common activity in the brain to the position of interest [24]. The main idea here is to remove the averaged brain activity which can be seen as EEG noise. The formula to calculate CAR is:

$$V_i^{CAR} = V_i^{ER} - \frac{1}{n}\sum_{j=1}^{n} V_j^{ER}. \tag{3.1}$$

where V_i^{ER} is the potential between the i^{th} electrode and the reference and n is the number of electrodes recorded.

3.1.2 Detrending

Detrending is a mathematical or statistical tool for removing trends from the data. Detrending is very efficiently used in preprocessing to prepare a data for analysis by methods that assumes stationarity. Trends can be defined as slow and gradual change in the statistical property of the process under the whole interval under investigation. It is also sometimes defined as long term change in the mean of the process but it could refer to other statistical properties also. Detrending has same effect on the frequency spectrum as that of a high pass filter, which means the variance at the low frequencies is diminished in comparison to the variance at high frequencies [25].

In our study we would be using Autoregression (AR) modeling for feature extraction which assumes a data to be stationary, which means a data to AR model has to strictly follow the properties of Wide Sense Stationary (WSS) signals.

A WSS signal has the following properties :

- The mean value of the process X(t) is a constant i.e,

$$E[X(t)] = m = constant \tag{3.2}$$

- Its autocorrelation R_{XX} function depends only on τ and not on $\boldsymbol{t}$ i.e,

$$E[X(t)X(t+\tau)] = R_{XX}(\tau) \tag{3.3}$$

We have use linear detrending for our system. Linear detrending calculates the best linear fit (trend) for the signal and then subtracts it from the signal to obtain a detrended signal.

$$DetrendedSignal = Signal - trend \tag{3.4}$$

3.2 Feature Extraction

This phase involves extracting those features of the signal that display certain characteristic properties of EEG signal that are unique to the signal and are thus suitable for the classification purpose. Extracting these features also reduces the amount of data that is fed to classifier and thus reduce the processing time of the BCI system. The features that are generally used for classification include FFTs, the PSDs, The Auto Regressive Coefficients, The Multivariate Autoregressive Coefficients and The Time- Frequency transforms (like the Wigner Ville transform).

The feature extraction should be performed in such a way that the task of classifying the EEG signal becomes as easy as possible. In order to achieve this, a prior knowledge about the changes that occur in the EEG signal during planning and execution of movements is used to select the type of features that best describes these changes.

This section explained the feature selection method used for the study and also the estimation of various parameters required for it.

3.2.1 Selection of Method

The classical approach to feature extraction in BCI is to estimate the power at the carefully chosen frequency bands in Fast Fourier Transform (FFT) generated spectra [26] [5] [10]. But recently AR spectral analysis has been widely used in BCI applications [27] [28] [26] [29]. AR spectral analysis is preferred over FFT by many researchers [26] [30] [31]. AR has following advantage in comparison to FFT technique in the field of BCI: firstly, AR modeling provides a better frequency resolution, this is due to the fact that AR estimation is a function of continuously varying frequency so there is no special significance to specific equally spaced frequency as there in FFT [26]. Secondly, with AR good spectral can be obtained even with short EEG segments. This is very important feature in regard to BCI as shortening the time

segment will enhance the speed of the system which in return provides a quicker classification rate in case of real time BCI [26] [30].

All these findings have motivated us to choose AR modeling for our study as feature extraction method. Now, the decision is to select the specefic AR technique. Two of the most common approaches for AR modeling is a simple fixed AR Model and an Adaptive AR model (AAR). The AR model is only capable of modeling stationary signals, but usually the signals for which AR models are used are non-stationary. In order to be able to do so, there are possible methods: first, to segment the signal into stationary segments of fixed length and perform AR modeling on these individual segments. Secondly, to use the modified version of AR that is AAR, AAR is capable of handling a non-stationary signal and for each instant of time estimates the AAR model based on the present information and past information of the signal.

However, many researches have preferred AAR over AR when it comes to real time due to their adaptive nature [27] [28], but it has been shown that if the data segment to AR model can be made stationary for sure, then AR model can perform better than AAR [32]. In order to achieve the stationarity for AR model, we decide to include a detrending module prior to AR modeling which made sure that the data received by AR module is stationary, refer subsection: 3.1.2.

Hence, we decided to choose AR modeling as feature extraction tool for our study.

3.2.2 AutoRegression Modeling

Autoregression is also known as Infinite Impulse Filter (IIR),as there exists memory or feedback in this system. It is also known as an all pole system.

The equation describing an basic AR filter is :

$$X_t = \sum_{i=1}^{P} A_i X_{t-i} + \epsilon_t \tag{3.5}$$

where A_i are the autoregression coefficients, X_t is the series under investigation and P is the order of the filter and ϵ is the noise term.

There are various methods to calculate AR coefficients, the two most common methods are Burg's method and Yule-Walker method. Earlier research have shown that Burg's method gives a better classification result in comparison to least square method for BCI [32] [33]. It is due to this reason that Yule-Walker method does not work efficiently with short time sequences as it requires more data points relative to model order of the system. On the other hand, Burg's method has shown good results with short time sequences with the only restriction that model order should be carefully selected [33].

Based on the information gathered from the above mentioned studies we decided to choose Burg's method for our study. The TSA AR Spectrum VI of LabVIEW uses Levinson-Durbin

Algorithm for calculating AR coefficients for Burg Lattice method, for Levinson-Durbin Algorithm and more details on AR please refer Appendix: C

3.2.3 Selection of Order

While working with AR modeling, it is very important to have correct values of A_i for a given X_t, and this can be assured by using a proper order for the system. If the selected model order is too low it will cause low resolution and loss of information where as too high model order will give spurious spectral results with lot of noise and unwanted information [33].

One way to test the order of system is to plot Function Prediction Error (FPE) or Akaike Information Criterion (AIC) versus order of the system. The curve will initially show an exponential decrease and becomes flat after a particular order. The order after which response becomes flat or consistent is chosen as appropriate order, in other words the model order is chosen which minimizes the following functions given by eq: 3.6, eq: 3.8 .

A Functional Prediction Error is defined by :

$$FPE = S_P^2 \frac{N+P+1}{N-P-1} \tag{3.6}$$

where N is number of data points, P is the order of filter and S_P^2 is the total squared error divided by N.

$$S_P^2 = \frac{1}{N} \sum_{P}^{N-1} \epsilon_t^2 \tag{3.7}$$

And Akaike Information Criterion is given by :

$$AIC = NlnS_P^2 + 2P \tag{3.8}$$

Both FPE and AIC gives very much similar result while plotting against order of the filter [34]. We have used FPE in order to select the order for our system.

The various tests and results for the model order selection can be found in Appendix: C

3.2.4 Maximum Entropy Method for Power Spectral Density estimation

The AR coefficients obtained from the AR model were used to generate the Power Spectral Density (PSD). The Maximum Entropy Method (MEM) uses an all pole AR model to calculate the PSD for the data sequence X(t) [33]:

$$S(f) = \frac{\sigma^2}{df|\sum_{k=0}^{N-1} A_k' exp - \frac{j2\Pi kf}{f_s}|^2} \tag{3.9}$$

It can be calculated by using eq: 3.7, where S(f) is the PSD of the time series. *df* is the frequency interval or the frequency resolution, which is computed as *fs/N*. *N* is the number of frequency bins, *fs* is the sampling rate, and σ^2 is the noise variance of the estimated AR model of the time series. *A* is an array that contains the coefficients of the AR model. A=[1, A1, A2,...,An], where *n* is AR model order. This array *A* is then zero-padded by *(N-n)* times zeros for form $A^{'}$.

For our study, we have used *N = 125* in order to achieve the frequency resolution *df* of *(fs/N) = (250/125) = 2* Hz. The obtained Spectral information from the MEM filter is used as the feature space for the proceedings modules. It is represented as the PSD value for each frequency bin and its corresponding channel. The number of frequency bins used to represent the PSD is *(N/2 + 1)*, i.e *63* as the other half will carry the same information as that of first half.

3.3 Feature Selection

The number of features compared to the number of samples must be adequate, as too many features in relation to the number of samples can produce over-fitting problems when classification is performed [35]. Hence, it is important to perform feature selection before conducting classification.

The feature selection is divided into two steps, first step is it to select good representative features to distinctive between classes. This is done to decrease the computational time and ignore features with irrelevant information. This selection can either be based on the information in the literature or statistical analysis tools such as R Square. The theory behind R Square is explained later in section: 3.5.1.

Second step is to project the data on to a one dimensional plane so that it can be classified using a linear classifier such as Bayesian.

3.3.1 Fisher's Linear Discriminant (FLD)

FLD is a classification method that projects high-dimensional data onto a line and performs classification in this one-dimensional space. The projection maximizes the distance between the means of the classes while minimizing the variance within each class [35].

The aim is to classify two different tasks, and FLD helps in finding a line that best divides the two tasks and minimize the error to least.

Considering an example where we have set of n d-dimensional samples x_1,....x_n, *n1* in the subset ω_1 and *n2* in the subset ω_2. A linear combination of the components of x can be written as :

$$y = w^t x \tag{3.10}$$

A corresponding set of n samples y_1,....yn divided into subsets Y_1 and Y_2. If $||w||$=1, then each y_i will represent a projection of corresponding x_i on to a line in the direction of w. The magnitude of w is not of any real significance but it is important to have correct direction for w.

And a best direction for w is one for which the ratio of between class separability to within class variability is maximum.

$$w = S_W^{-1}(m1 - m2) \tag{3.11}$$

where *m1* and *m2* are means of class1 and class2 and S_W is the covariance matrix for the within class.

Hence, FLD reduces the multi-dimensional feature space into one dimension. However, if the feature space can be reduced in more than one dimension that is 1 or 2, classification accuracy can be increased. However care shall be taken in increasing the number of dimensions, as the optimal number of dimensions is a trade off between increasing the classification accuracy and over-fitting the classifier. Many dimensions induce many features, including irrelevant and redundant features, which can result in a confuse training of the classifier and thereby disguise the function of the relevant features [35].

For more information please refer [35].

3.4 Classification

In BCI, it is very common to use the linear classifier and this is due to their simplicity of application. Linear classifiers are generally more robust than their nonlinear counterparts, since they have only limited flexibility (less free parameters to tune) and are, thus, less prone to over-fitting [3] [36].

We have used a Bayesian classifier on the reduced feature space. The Bayesian classifier a probabilistic approach to the pattern classification where the classification is independent of the specific training samples but instead depends on the class conditional probability.

The Bayes formula can be represented as:

$$posterior = \frac{likelihoodXprior}{evidence} \tag{3.12}$$

or

$$P(\omega_j|x) = \frac{p(x|\omega_j)}{p(x)} \tag{3.13}$$

There are many different ways to represent pattern classifiers. One of the most useful is in terms of discriminant functions $g_i(x)$, i= 1,.....c. The classifier is said to assign a feature vector x to class ω_i if

$$g_i(x) > g_j(x) \qquad for\ all\ i \neq j \tag{3.14}$$

When features overlap in a Bayesian classifier, it selects the class having the highest posterior probability, and there by minimizing the error. The discriminant functions $g_i(x)$ defining the classifier are therefore the posterior probability, which can be represented as:

$$g_i(x) = P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_i)}{p(x)} \tag{3.15}$$

Since p(x) is the same for all classes and the a priori probabilities $P(\omega_i)$ are the same for both classes in our study, the problem was reduced to finding $p(x|\omega_i)$. The conditional probabilities can be found by assuming a Gaussian distribution of features, in which case the standard expression for the multivariate normal distribution can be used:

$$g_i(x) = \frac{1}{(2\Pi)^{\frac{d}{2}}|\Sigma^{\frac{1}{2}}|} exp[-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)] \tag{3.16}$$

where x is the feature vector of d dimensions μ_i and Σi are the mean vector and covariance matrix for the i^{th} class.

Using the above discriminant function we can obtain the equation for minimum rate classification, which can be expressed as :

$$g_i(x) = ln\, p(x|\omega_i)\ +\ ln\, P(\omega_i) \tag{3.17}$$

This expression can be readily evaluated if the densities $p(x|\omega_i)$ are multivariate normal that is, if $p(x|\omega_i) \sim$ N(μ_i,Σ_i). In this case, 3.16 can be solved as :

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^t\Sigma_i^{-1}(x-\mu_i)\ -\ \frac{d}{2}ln2\Pi\ -\ \frac{1}{2}ln|\Sigma_i|\ +\ lnP(\omega_i) \tag{3.18}$$

or 3.18 can be re-written as:

$$g_i(x) = x^tW_ix\ +\ w_i^tx\ +\ w_{i0} \tag{3.19}$$

where

$$W_i = -\tfrac{1}{2}\Sigma_i^{-1} \quad , \quad w_i = \Sigma_i^{-1}\mu_i \tag{3.20}$$

$$w_{i0} = -\frac{1}{2}\mu_i^{\ t}\Sigma_i^{-1}\mu_i\ -\ \frac{1}{2}ln|\Sigma_i|\ +\ lnP(\omega_i) \tag{3.21}$$

In this manner, we have reduced our calculation to mean and covariance vectors and using them we can train our classifier. And using this mean and covariance calculated g_1 and g_2 in

order to have a classification result.

Based on their values if $g_1 > g_2$, then the result is classified as class 1 (left) in other case it is classified as class 2 (right). However, we have also used the values of g_1 and g_2 in order to calculate g_{error} , which is saved as a one of the States.

$$g_{error} = g_1 - g_2 \tag{3.22}$$

The normalised value of g_{error} is used in the feedback display. The feedback display has a axis of +1 to -1 representing left and right respectively. Using the maximum and minimum value of g_{error} from the Screening data, the positive half and negative half of feedback display is normalised respectively. The g_{error} computed during each second classification is then normalised using g_{error_max} or g_{error_min} depending on whether it is positive or negative and is displayed on feedback display. To have look at the feedback display, please refer section: 4.2.2.

For more details about the Bayesian Classifier, please refer [35].

3.5 Offline Tools

3.5.1 R Square

As mentioned, the first step of feature selection is to reduce the feature space as to remove irrelevant information, in order to achieve the same we have implemented R Square operation on whole feature space.

R Square also known as coefficient of determination is defined as variability in the data set that is due to the statistic model. R Square can be described in two forms depending on the number of independent variables.

R Square or coefficient of simple determination is defined as the percentage of variance in the dependent variable that can be explained by the independent variable.

$$r^2 = \frac{SS_{Regression}}{SS_{Total}} \tag{3.23}$$

And R Square, the coefficient of multiple determination is defined as the percentage of variance in the dependent variable that can be explained by taking all the independent variables together, which is the case in our project,

$$R^2 = 1 - \frac{SS_{Error}}{SS_{Total}} = \frac{SS_{Regression}}{SS_{Total}} \tag{3.24}$$

where *SS_Error* is the sum squared of errors, *SS_Regression* is the sum squared of regression and the *SS_Total* is the sum squared of total variation in variable,where.

$$SS_{Error} = \sum(y_i - \bar{y})^2, \quad SS_{Regression} = \sum(\hat{y}_i - \bar{y})^2, \quad SS_{Regression} = \sum(\hat{y}_i - \bar{y})^2 + \sum(y_i - \bar{y})^2 \tag{3.25}$$

Hence, in our case R Square helps in determining the features that best describes the two different classes, depending on the desired number of features the topmost values from R Square module is used in the future module.

The R Square module in our program is implemented on the basis of matlab code provided by Marco [37] and Alvaro [17].

3.5.2 Cross Validation

Previous section provides a explanation about the classification technique used, but the classifier cannot be trusted unless it is being tested for its performance. Therefore, in order to test the performance of the classifier, we implemented cross validation in the offline analysis to validate our classifier.

Cross-Validation is a technique where the data is divided into several subsets, where one subset is used as the testing subset and the rest all are used as training subsets.

For our project, we have K-Fold Cross-Validation technique, where initially the data is divided in to K number of subsets each of equal size. Of this K subsets, one is selected for validation or the testing and the rest (K-1) are used as the training subsets. The same process is repeated K times, with each of K subsets used as the testing data only once. The K results obtained from the Cross Validation are then added in order to get a single estimation of all the data.

In our case, we have used 10-Fold Cross Validation.

Chapter 4

Experiment Protocol

This chapter provides the detailed description of the experimental procedure followed as well as the parameters which are vital in consideration to the experimental design.

4.1 Subjects

2 subjects both males participated in the experiment as volunteers. They reported no history of neurological disorders.

4.2 System Set-Up

The subject was made to sit in a reclining chair looking at the center of computer screen kept at 90 degree to table and 100cm from them with a straight head as shown in figure: 4.1. The whole experiment was conducted with no power in the room and with The BCILab program to record the sessions running on a laptop powered by battery. Only the external screen for the subject to look at was powered and that with a cable from the adjacent room connection in order to avoid any electrical interferences. Also, it was made sure that subject was not distracted by any other sources of external noise.

An elastic Electro-Cap cap was placed on the scalp and used to record EEG signals through 38 channels. The arrangement of the cap was according to an extended 10-20 system of electrode placement as shown in the appendix: F figure: F.7. The channels on the cap were referenced to the electrically linked earlobes, A1 and A2. These were cleaned to remove dead skin cells before connecting the electrodes. The impedances of all the electrodes was maintained at less than $5k\Omega$ using an electro gel. 6 EMGs channels were added in order to detect any physical movement due to the hand movement or the eye. Two pairs of electrodes were placed on left and right forearm, acting one as reference. Another two were placed near the left eye, one placed on the chick bone and the other right adjacent to the eye to record the vertical as well as the horizontal movements of the eyes. All the EMG electrodes were grounded to an electrode connected on leg where as the EEG electrodes were grounded to one of the cap

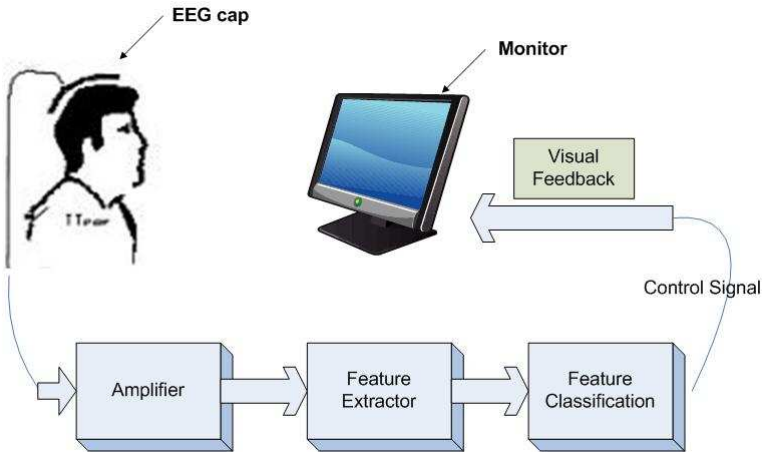

Figure 4.1: A simple arrangement of BCI System. Figure partially adopted from [3].

Figure 4.2: Subject sitting in the Setup.

electrodes (Electrode GND). The complete arrangement of the system can be seen in the figure: 4.2. Further details can be found in appendix: F.

The data was acquired using the ACQUIRE program in conjunction with the BCILab program. ACQUIRE is a part of Scan 4.3 package provided by the Neuroscan. The NuAmp amplifier by Neuroscan was set to a high pass filter value of 0.5 Hz and a notch filter at 50 Hz together with a sampling rate of 250 Hz and a gain of 19. The data acquired by ACQUIRE was transferred continuously to BCILab using a TCP connection. BCILab stored all the data acquired by ACQUIRE together with BCILab parameters, States and Subject information. For the information regarding the BCILab setup parameters please refer chapter: 5 and chapter: 6.

The experiment was divided into sessions, one screening session followed by feedback sessions over the following days. The subject was asked to imagine a movement with his finger as if he was tapping a mouse button both during the screening as well as during the feedback sessions. After starting of the sessions the subject was left alone in the room. The following

sections will take each session in detail.

4.2.1 Screening Session

Screening Session was aimed at collecting the data in order to train the classifier. Hence, a screening session was done once for each subject at the start of the study. The model generated from the screening data was used as a training data in further feedback sessions.

The screen display for the subject during the screening session can be seen in figure: **??**. The subject was given an oral warning for the start of experiment. The tank filling display, called Time Before Screening was a visual information for the appearance of target. As soon as the tank was filled to half of its capacity, either the indicator for target left or target right would light up. The total time for the tank to get filled was 10s and as soon as the tank got filled, subject started performing his task. Each trial ran for 10s during in which subject performed the task while the EEG signals got processed and the feature spaces saved every second. Then subject got a break of 7s before the start of the next trial. For a single session, a total number of 10 trials are made, where each trail was 10s long run. Hence, a single recording session had 100s of feature spaces.

After the completion of the screening session, the model from the recorded data was generated using the Feature Selector program found in the BCILab.llb and was saved as a .txt which was later used in the feedback session. We have used 6 features for our classifier, from C3 and C4 at 8, 10, and 12 Hz. The selection of these particular features was based on the literature which had proved that left and right imaginary finger movement can be classified using C3 and C4 at the mu rhythms [38] [13] [39] [40].

4.2.2 Feedback Session

The feedback session had a similar setup as that of the screening, the difference was during feedback session subject received the visual feedback about his performance. The screen display for the feedback session can be seen in figure: **??**.

The Error Meter provided the feedback information about the performance of every second. The Success Rating indicator accumulated the number of classified lefts against number of classified rights during a single 10s trial. When the trial was over the value of Success Rating gave the result state and the target indicator for the class found in majority would blink during the reward time (5s). In the case of equal amount of classified lefts and rights none of the indicators would blink. The Error Meter, Success Rating and result code were reset to zero after every trial.

During the feedback session data was recorded for offline analysis too, so, if desired, the feature spaces could be examinated. Feedback sessions also had 10 trials of 10s each.

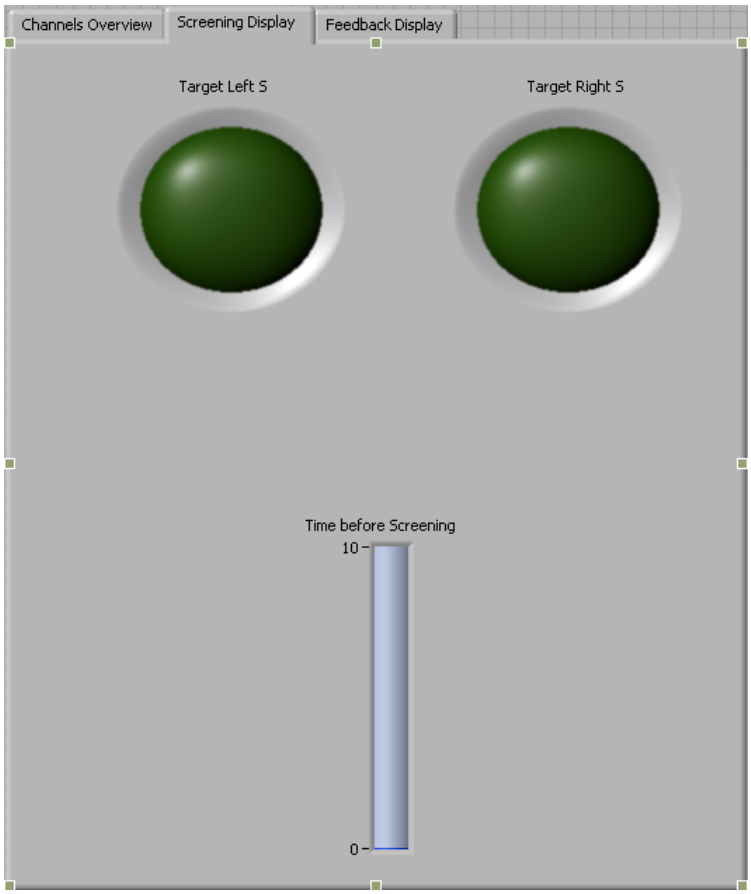

Figure 4.3: Screening Display

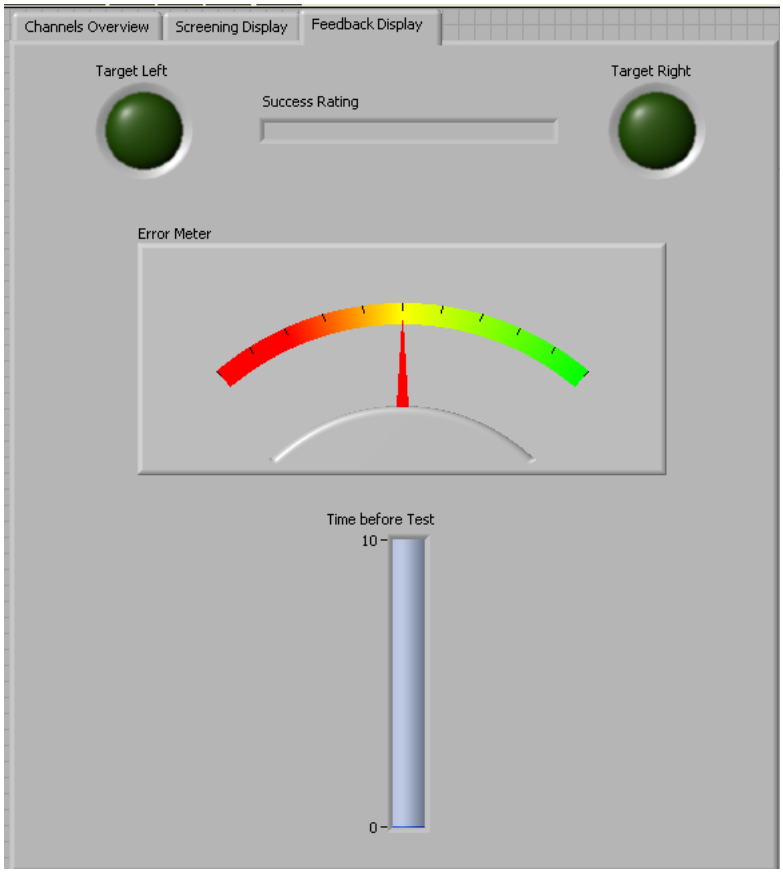

Figure 4.4: Feedback Display

Note: The BCILab program and data recorded during the study can be found in attached CD and at the BCI server respectively.

Chapter 5

The BCILab Online Program

The Online program is called BCILab.vi and can be found in the BCILab.llb library file. It is made on a basis of another program made by Clemens Eder [41] which interacts with the ACQUIRE program through TCP. Clemens program was meant for monitoring a single EEG channel and contained no options for selecting multiple channels, storage of data or control of sessions. This we had to implement. What we did get from Clemens program was the TCP setup and data receiving tools. For our program to run it has to be on the same computer as ACQUIRE.

The program has 3 modes decided by which page is visible on the Display Tap when starting the data acquisition. The mode can not be changed during runtime. The pages on the Control Tap can always be flipped through, also during run time. All parameters has to be set though before acquisition start. This start happens when the Start Acquisition button is pressed. Naturally, parameters can also be set before the program is even run. The only option that has to be set before program run to work is the selection of .asc-file. This is not required for the program to run though. Starting of the ACQUIRE server should be done before program run, remember to set the same port in both programs. After start of acquisition the program should be stopped again by pressing the STOP button, this will close down the TCP connection properly. If the program gets terminated with Abort Execution or anything similar the ACQUIRE server has to be restarted.

In short is the program operated by first setting and selecting all parameters. Then the program is executed by pressing Run. Any forgotten parameters can still be set until the Start Acquisition button is pressed, and by selecting a page in the Display Tap the program mode is chosen. To start a screening or feedback session the operator has now to push the Start Test button. The test will automatically run its course and ends with the Start Test button going back to its default position ('false'). The program can at any time be ended by pressing the STOP button.

5.1 The Control Tap

These pages contains mostly the controls for the parameters of the program. They also contains the buttons to control the program flow. Lastly they contain some indicators that are useful for the operator of the program, but that the subject should not be bothered with during sessions. Most parameters are important during screening and feedback sessions, but not when using the Channels Overview.

5.1.1 Main Controls

Figure 5.1 shows the page for the Main Controls.

- A: The TCP port number. Has to match the one in ACQUIRE. Our selection is 4001 but can be any available.
- B: The size of the data buffer where the received samples are stored. During screening and feedback sessions this buffer should be at least as long as the sampling frequency, but it is no problem if it is longer. When using the Channels Overview mode this array will be plotted in the Channels Display. The number of seconds plotted will then be equal to the size of this data buffer divided by the sampling frequency. Our selection is 1000.
- C: Total number of recorded channels. This value is provided to the program through an inquiry to ACQUIRE after program run.
- D: Sampling frequency. Also this value is provided by ACQUIRE.
- E: This operator is the only left unknown operator from [41] program. Apparently the value has to be one.
- F: Gives the current count of samples in each channel stored in the data buffer. This number is the same for all channels. During screening and feedback sessions it will return to zero after it has reached the value of the sampling frequency indicating that the data has been processed and saved and a second has passed.
- G: The name of the subject.
- H: The current session number for the subject.
- I: Name of the .dat file where program parameters and the raw data should be saved. If the file already exist the program will prompt the operator when starting the acquisition.
- J: Path to the training data parameters generated by the Offline Tools.
- K: Path to the .asc channel name information file. This file can be generated in ACQUIRE by exporting the Channel Layout. The path should be set before program run to work.

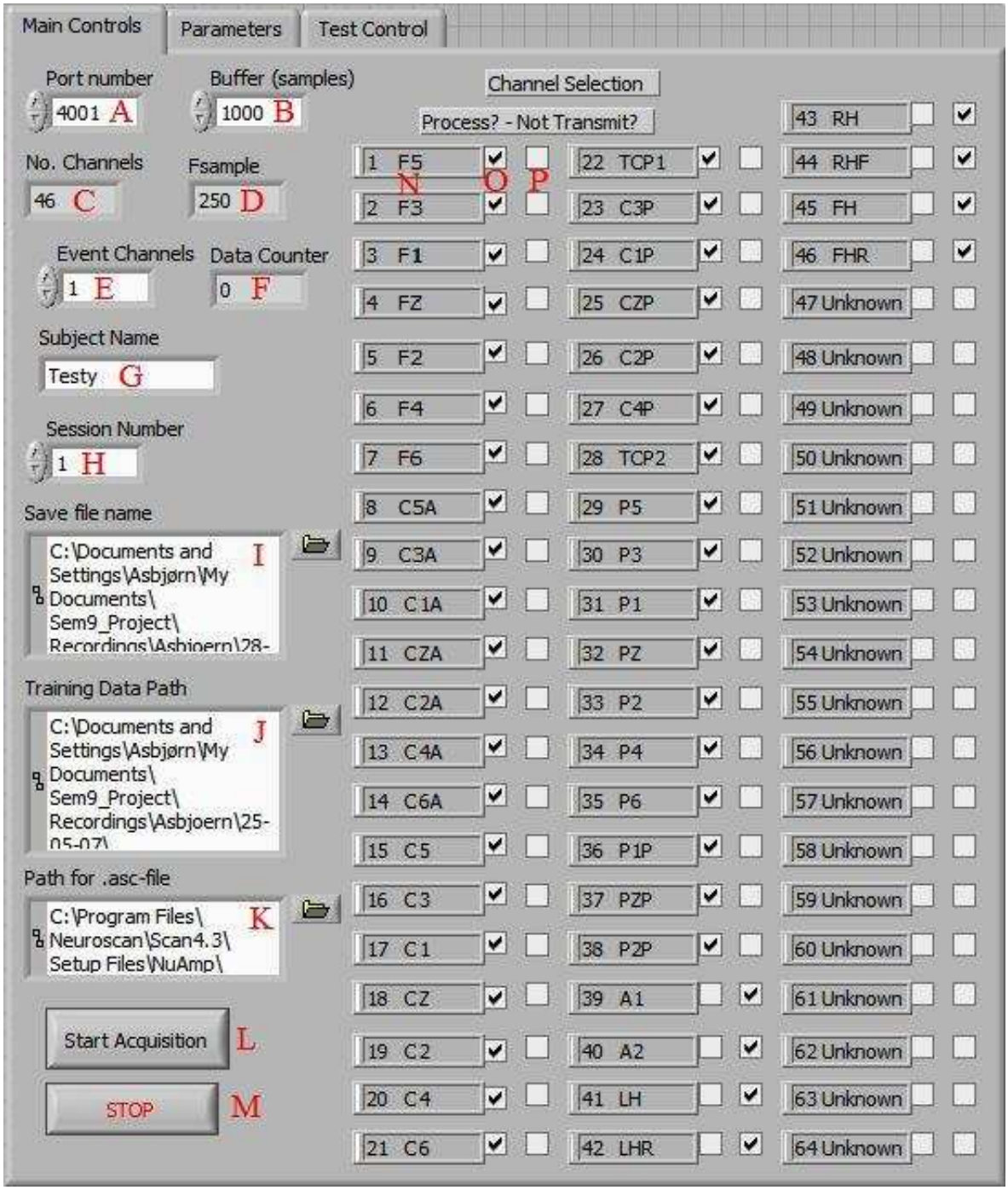

Figure 5.1: The "Main Controls" page.

- L: Sends a command to ACQUIRE to start sending data while at the same time ends the program setup phase. If the program mode is either screening or feedback all information about parameters and setup will be saved in the .dat file. If the program mode is Channel Overview no data will be saved.
- M: This button stops the program and ends it. It will only work after Start Acquisition has been pressed.
- N: If a .asc has been selected before program run these boxes will show the name of the channels. If a file has not been selected or there are more channels on the page than names, the boxes will just show the channel number followed by 'Unknown'. This has no effect on if a channel can be recorded and processed or not.
- O: The tick boxes in this column indicates if a channels should be run through the signal processing or not. For screening and feedback sessions all EEG channels should be selected. In Channels Overview mode will the selected channels be the only ones shown on the Channel Display.
- P: The tick box in this column indicates if the channel should pass through the transmission filter or not. If a channel does not go through the transmission filter it will not be used in the detrend calculations. All non-EEG channels should be ticked when doing screening and feedback sessions. The option has no influence on the Channels Overview.

5.1.2 Parameters

These controls are only of effect during screening and feedback sessions. They control the signal processing modules we have implemented. Figure 5.1 shows the page for the Parameters.

- Q: The voltage offset for all channels. Our selection is zero.
- R: The voltage gain for all channels. This value is dependent on the amplifier and, in the case of the NuAmb, always be 19, which is also our selection.
- S: The bit to micro Volt conversion ratio. After the program has received the data samples it has to convert it into micro Volt before saving it as raw data in the .dat file. This is following the BCI2000 standard. The conversion value can be found in the ACQUIRE program on the Amplifier page in the "Overall Parameters..." option. Our selection is the fixed value for the NuAmp; 0.063 uV/LSB.
- T: Sets how many seconds the two tank indicators Time before Screening or Time before Test should fill up before the subject has to focus on the chosen target. The target state will be randomly chosen after half of this time. Our selection is 10 seconds.
- U: Sets the total number of trials for the session. That means the total number of times the subject will be presented with a new target to hit. Our selection is 10 trials.

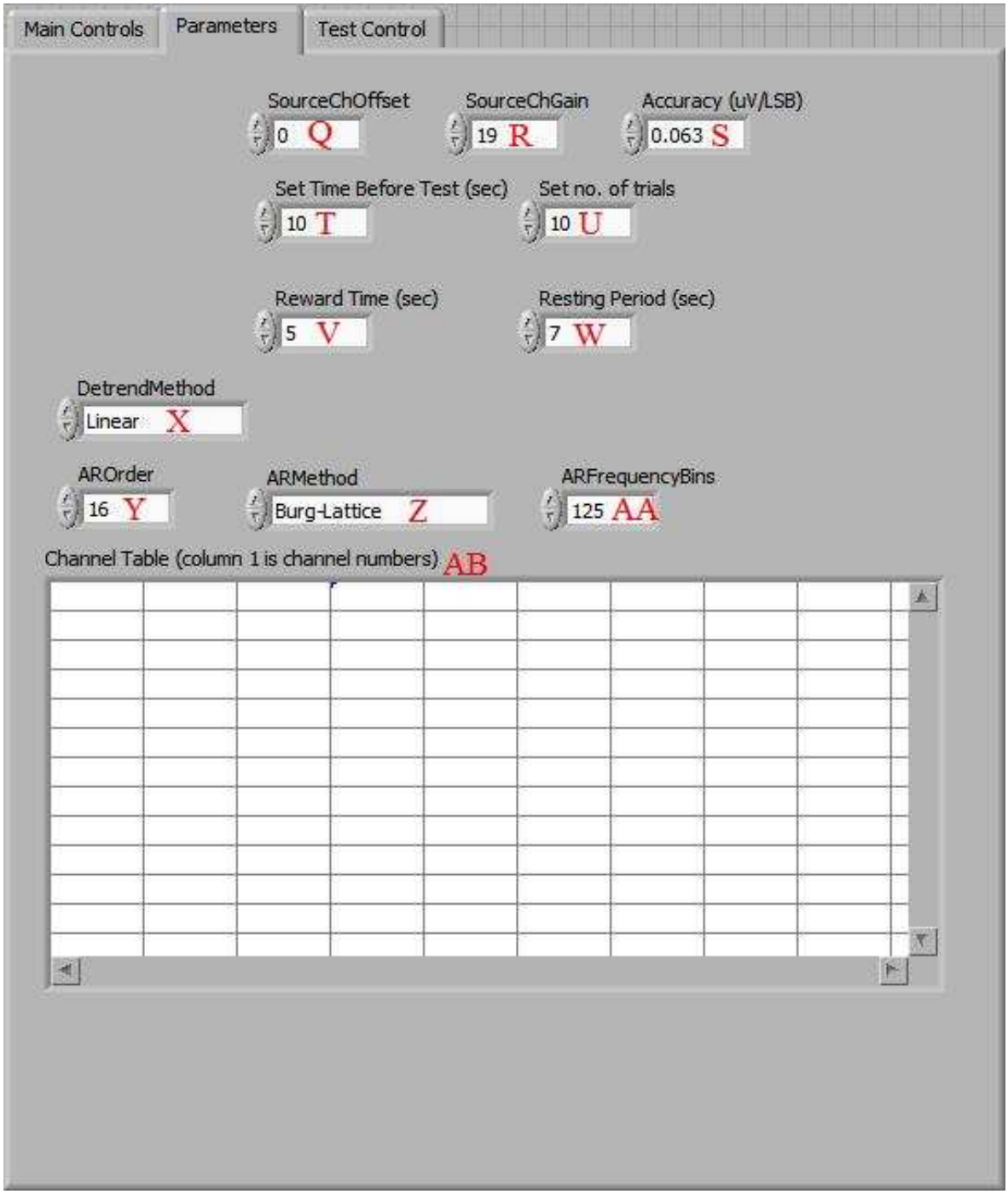

Figure 5.2: The "Parameters" page.

- V: During a feedback session this option decides for how long the resulting target indicator should blink. Our selection is 5 seconds.
- W: The time for the resting period between trials. Our selection is 7 seconds.
- X: Selects the method for our detrend module, see 3.1.2 for more info. Our selection is Linear.
- Y: The order of our AR module, see 3.2.3 for more info. Our selection is 16.
- Z: The method for our AR module, see 3.2.2 for more info. Our selection is Burg-Lattice.
- AA: The number of desired frequency bins in the AR module, see 3.2.4 for more info. Our selection is 125 bins.
- AB: During feedback sessions the feature bins used for classification is displayed here. The bins are loaded from the training data file. The layout is similar to the Channel Table described in the Offline Tools, see 6.1. It is not advised to change values in the table manually.

5.1.3 Session Control

This page shows indicators of progress during screening and feedback sessions. As not to have the subject distracted by the indicators it is advised to scroll the front panel to the right so only the Display Tap and the Start Session button is visible. All state parameters get saved as states in the .dat file every second. Figure 5.1 shows the page for the Session Control.

- AC: State parameter. Every time the data buffer has received the sampling frequency number of samples it is assumed that one second has passed and this indicator updates. It start at zero when Start Acquisition is pressed.
- AD: Gives the number of the current trial.
- AE: The number of trials left until the total number of trials for the session is reached.
- AF: During a feedback session this indicator shows how many of the targets that were classified correct by the end of the trial.
- AG: State parameter. Indicates if the program is running or not. Will always be 'true' during runtime.
- AH: State parameter. Indicates if data is being processed and stored in the feature space save file.
- AI: State parameter. Indicates the target code during the trial, 0: none, 1: left, 2: right.
- AJ: State parameter. Indicates the result code after the trial is over, 0: none, 1: left, 2: right.

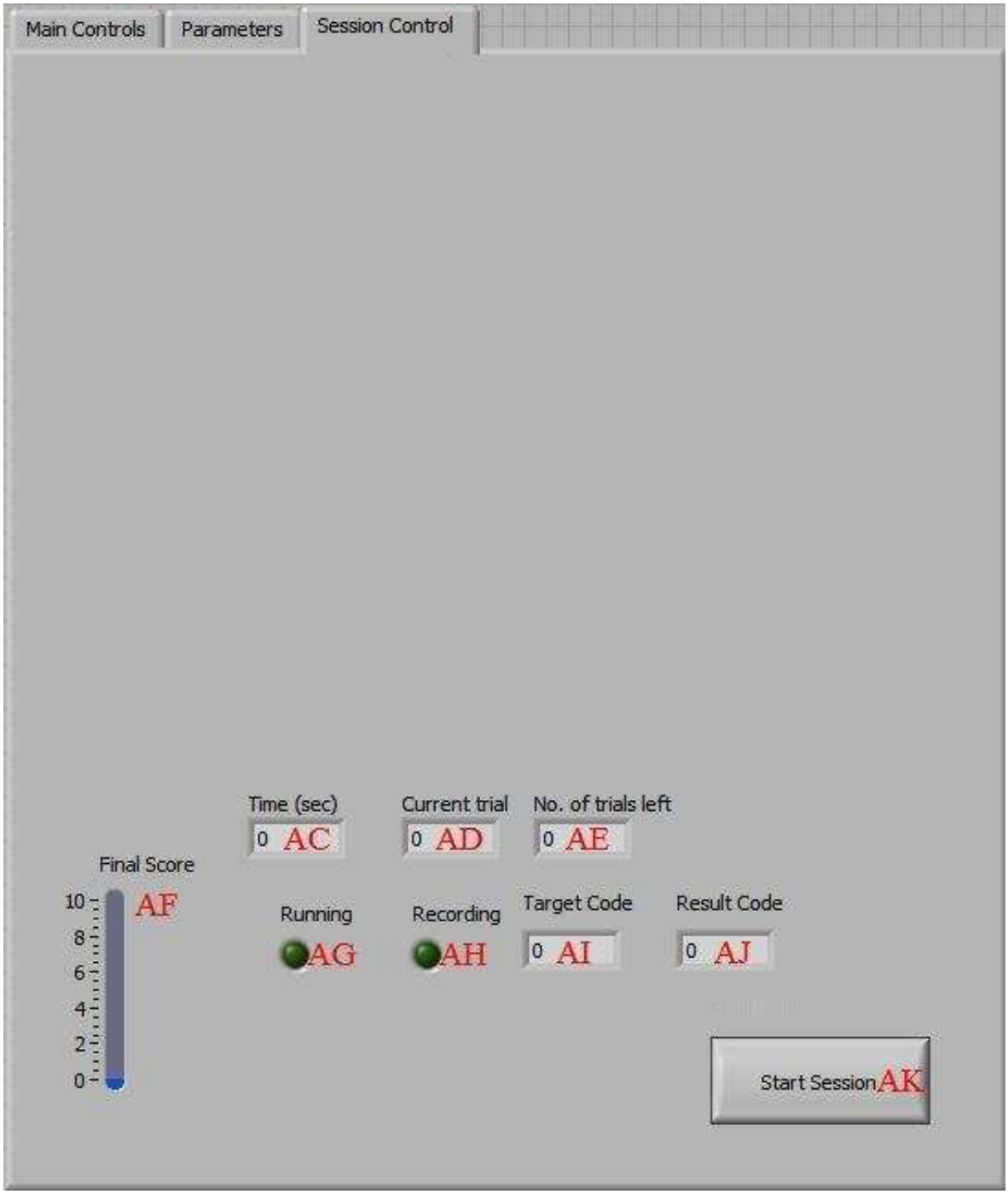

Figure 5.3: The "Session Control" page.

- AK: Starts the session. When the session is over the button will return to default ('false').

5.2 The Display Tap

The visible page on the Display Tap determines the mode of the program. During screening and feedback sessions is it important to center the Display Tap in front of the target. The program is made to always give the same amount of left and right targets but the order is random.

5.2.1 Channels Overview

This mode is useful to check if raw data is being received or not. It is also possible to display only a few selected channels for display unlike in ACQUIRE. Unlike the two other display pages does this one contain controllers which can both be changed during runtime. Figure 5.4 shows the page for the Channels Overview.

- AL: Displays the channels by plotting the samples in the data buffer. The channels will be plotted one after another with channel 1 in the bottom. Channels that have not been selected for plotting will be shown as a line.
- AM: Controls the distance between the plotted channels. By increasing it the channels be come more separate.
- AN: Multiplier for the plotted data. This multiplies the amplitude of the samples and works like a zoom in/zoom out controller.

5.2.2 Screening Display

The only information the subject gets during screening sessions is what target movement to imagine and when. Figure 5.5 shows the page for the Screening Display.

- AO: Indicators for the target movement to imagine. One of them will turn on when the Time before Screening is half filled up and will turn off when the trial enter the resting period.
- AP: Starts filling up at the beginning of the trial one element pr. second.

5.2.3 Feedback Display

The indicators on this page are similar to those on the Screening Display but with two more added: Error Meter and Success Rating. Both of these gives visual feedback to the subject during sessions about his performance. After a trial is over the accumulated result in Success

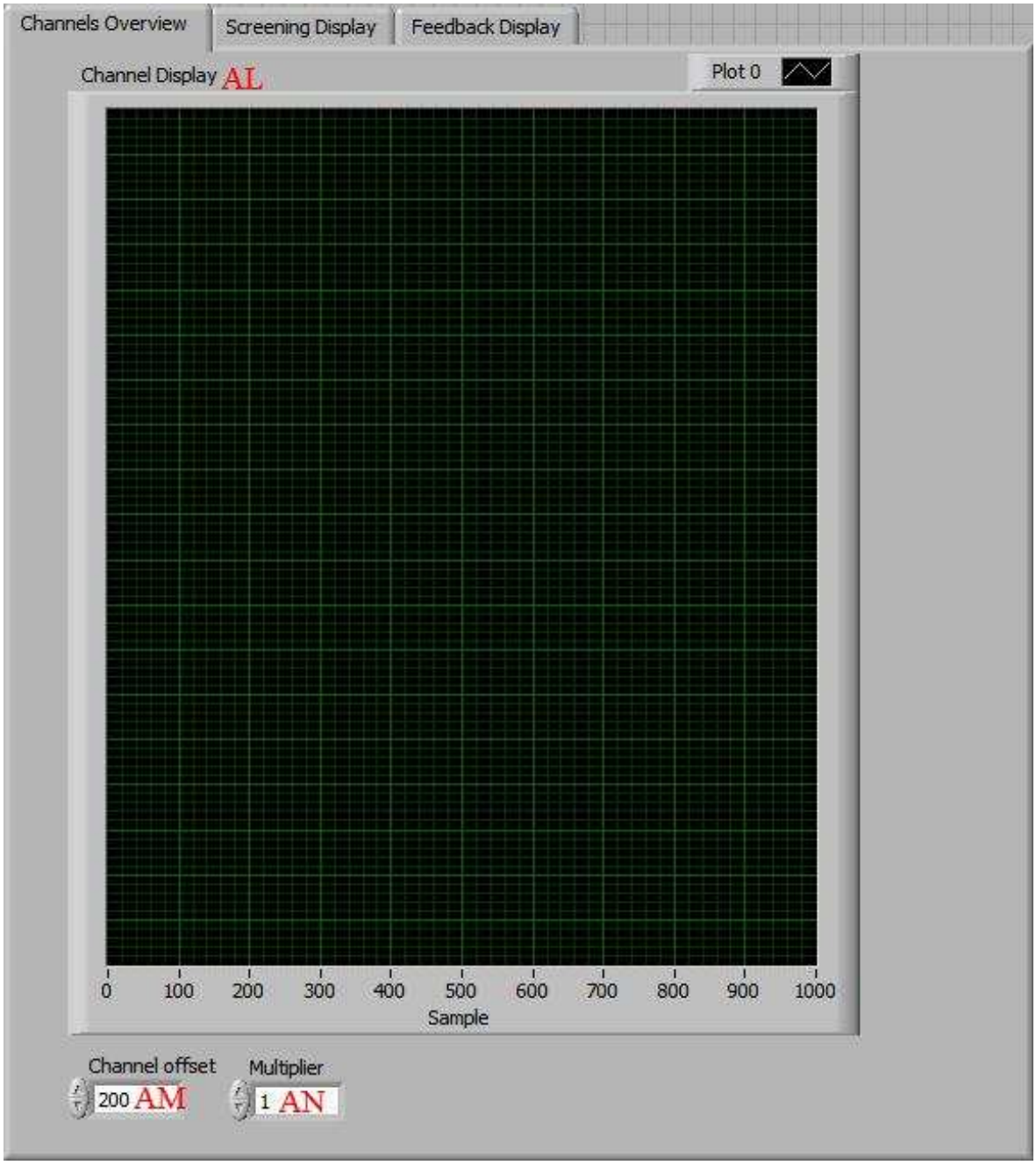

Figure 5.4: The "Channels Overview" page.

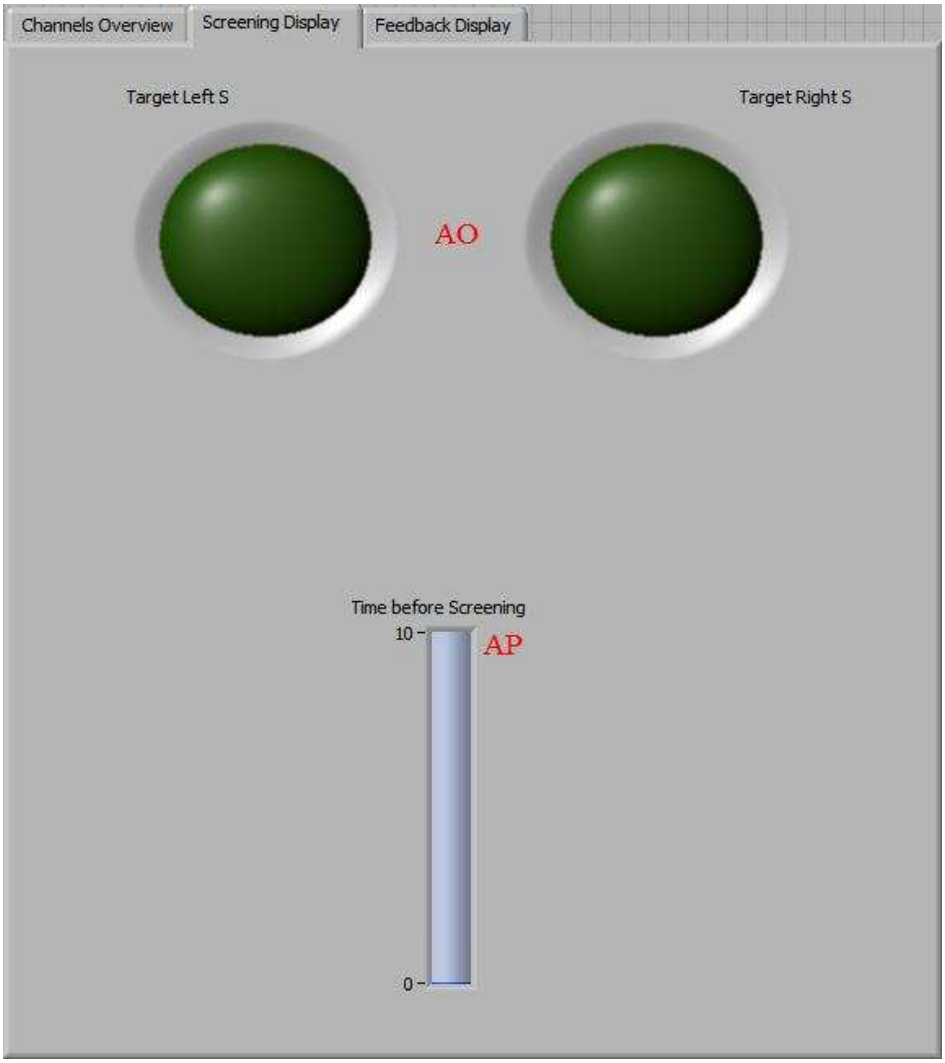

Figure 5.5: The "Screening Display" page.

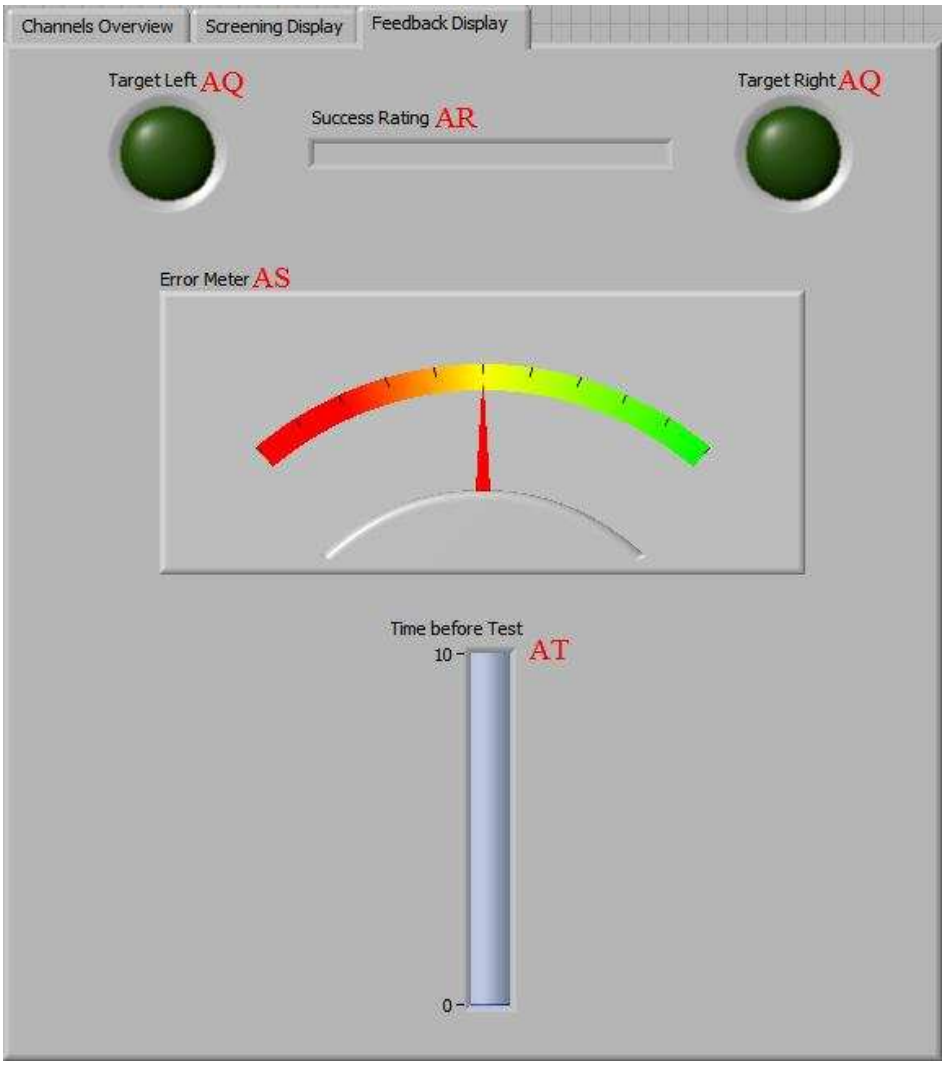

Figure 5.6: The "Feedback Display" page.

Rating will be evaluated. If it is positive to either of the targets that indicator will blink during the reward time phase and that target will be set as Result State. If the Success Rating is zero no indicator will blink and the Result State will be zero. All indicators reset when a new trial is about to begin. Figure 5.6 shows the page for the Feedback Display.

- AQ: Indicators for the target movement to imagine. One of them will turn on when the Time before Screening is half filled up and will turn off when the trial enter the resting period.
- AR: Accumulates the results of the classifier every second. It will fill up toward the target that gets classified. If the classification result shifts to the opposite target the fill will decrease to zero and then start fill up toward that target.
- AS: State parameter. Displays the g_{error} value of the last classification, see 3.4 for more

info. The indicator is updated every second.

- AT: Starts filling up at the beginning of the trial one element pr. second.

5.3 Data storage

We are trying to store the recorded data in a format according to the specifications of BCI2000 in .dat files. This is done to allow us use of the advanced off line analysis tools for BCI2000 for our data recordings. The storage files contains three elements: state definitions, parameters and binary data. The state definitions and parameters are saved during initialization of the LabVIEW program while the binary data is saved continuously during runtime. The binary data contains the sample values of all channels followed by the state values.

Sadly we encountered some unknown problems getting our .dat files read by the newest file reader module for BCI2000 called load_bcidat.m, instead our files can be read by another reader module called bci2k_readdat.m.

5.3.1 State definitions

The state definitions describes the name and bit size of the states that you want to save for every sample recording. The states are defined like this:

'name' 'length 'value' 'byte number' 'bit number' CRLF

The 'name' is of course the name of the state. 'length' defines the bit size. 'value' is the default value of the state. The 'byte number' and 'bit number' describes how it is located in the state bytes. This is done not to waste multiple bytes of saving space for different boolean or few bit states. Instead they are combined. As example 4 states: state1 is 4 bit, state2 is 1 bit, state3 is 8 bit (1 byte) and state4 is 13 bit. Their definition will look like this:

state1 4 0 0 0
state2 1 0 0 4
state3 8 0 0 5
state4 13 0 1 5

This will give a total of 3 bytes and 2 bits for storing states. This has to be rounded up to nearest byte, so 4 bytes in all. When saving the states in the binary data it is important to keep their order correct.

Our saved states are:

- SourceTime, 16 bit. Time in seconds since acquisition start.

- Running, 1 bit. If program is running.
- Recording, 1 bit. If signal processing is being done and feature space saved.
- TargetState, 2 bit. What the target state for the subject is during trial, 0: none, 1: left, 2: right.
- ResultState, 2 bit. What the result state for the subject is after trial, 0: none, 1: left, 2: right.
- ErrorRatingSign, 1 bit. Sign of the g_{error} value, 0: plus, 1: minus.
- ErrorRating, 8 bit. Value of g_{error}.

As states are unsigned integers we had to save the sign for g_{error} in its own state. Also, as the floating point value for g_{error} often was below one we decided to multiply it with 100 before saving. The value saved has not been normalised for left or right but the normalisation factor can always be found in the training data file used during the feedback session.

5.3.2 Parameters

This is how you store the setup values for your recording. The format is:

'section' 'data type' 'name' 'value' 'default value' 'low range' 'high range' // 'comment' CRLF

The 'section' refers to which type of parameter this is. 'value' refers to the actual value of the parameter while 'default value' is not really of importance. 'low range' refers to the possible lowest value of the parameter and 'high value' to the highest. The comment describing each parameter can be read in the .dat file by opening it with a text editor. An example of the .dat file header containing state and parameter information can be found in appendix G.

5.3.3 Feature Space File

During sessions we save the feature spaces in a separate file. This is done so our Offline Tools can work on this file. The file is in simple text format and can be viewed in a text editor. It contains the values for all frequency bins on all selected channels for every second during a session computed by our AR module. Channels that have not been ticked for processing on the General Controls page will not be saved. Information about Target Code and the index number of the channel are also saved. Index zero is channel one.

5.3.4 Training Data File

The Offline Tools can generate these files which contains training data for the classifier in the online system system. If FLD is in use the values for this will be saved too. The file also contains scores for the best left and right g_{error} values. The file is in text format too.

5.4 Timing

As timing during sessions is the most important that will be the one described here. The timing of the system is controlled by the data acquisition from ACQUIRE. As there is no way to control the package flow through the TCP channel for our program it has to rely on ACQUIRE sending the number of samples per second in accordance with the sampling frequency. It has been tested and verified that ACQUIRE has a new data packages available 40ms. The number of samples in these packages increases with the sampling frequency ranging from 5 samples per channel when sampling frequency is 125 Hz to 40 samples pr channel when sampling frequency is 1000 Hz.

Three threads control the flow in BCI Lab. First thread handles the data receiving from ACQUIRE and stores it in the data buffer within the program. The thread has an idle time between executions of 30ms, a bit faster than ACQUIRE generates packages. This is no problem as the first thing the thread will do is execute a command to go to the TCP port and wait for package. The package should arrive 10ms later and gets removed from the channel. There is no chance of re-reading the same package twice as that is being prohibited by TCP. It should be noted that if the amplifier simulator mode is being used in ACQUIRE it will not generate a package every 40ms but every time a package is removed from the channel. The method of going to the port and wait gives BCILab an possibility to "catch up" if it is falling behind timing vise.

Second thread has an idle time of 10ms between executions. It will always check if the number of data in the data buffer equals the sampling frequency or not. If true, it will execute the signal processing and saving of the data and clear the data buffer. At the same time it will increase the Time counter for BCILab by one. As the threads idles less than the first thread it will have multiple opportunities to check the data buffer and empty the data buffer before the first thread adds more. A semaphore prevents access to the data buffer by both threads at the same time.

Last thread handles session execution and checks every 100ms if the Start Session button is active. If so it will execute the session protocol for either screening or feedback.

As explained here it does not seem possible in theory for BCILab to receive more samples than the sampling frequency before the data buffer is processed. Still, visual inspection of the stored data files indicates that some times one package gets too much received before processing. The frequency of this happening is little though, like less than one time out of

ten. Therefore you can easily have trials of recordings with out it happening at all. Also, as our AR module takes the sampling frequency into consideration it does not have much effect on our processing even if it should happen. (i.e. 250 or 260 samples sampled at 250 Hz does not do much of a difference.

Chapter 6

Offline Analysis Tools

The offline analysis tools we have developed can be executed by running the Feature Search.vi program in the bci.llb library file. The tools are used to examine and process the saved feature space files from the online program. It is also used to generate the training file needed to run the feedback sessions in the online program.

The front panel consist of some controllers for general options and a tap control that decides the program mode. An execution of the program will make it run one time and stop automatically. Only the page for the selected program mode will be updated during a run. All controllers need to be set before executing the program.

6.1 General Options

Figure 6.1 shows the controls for the general options.

- A: Path and file name of the feature space file that is wanted for processing.

- B: Selects if the feature space file should be loaded or not. As this can take some time the feature spaces will be stored in memory in the Feature Cluster Array object at marker G. If the same set of feature spaces have to be processed multiple times this option should be ticked after the first run.

- C: Selects if classification should be done with FLD or not. This parameter will get stored in the training file. Our selection is 'true'.

- D: The number of groups used in the Cross-Validation tests. The total number of feature spaces has to be a multiple of this number. Our selection is 10.

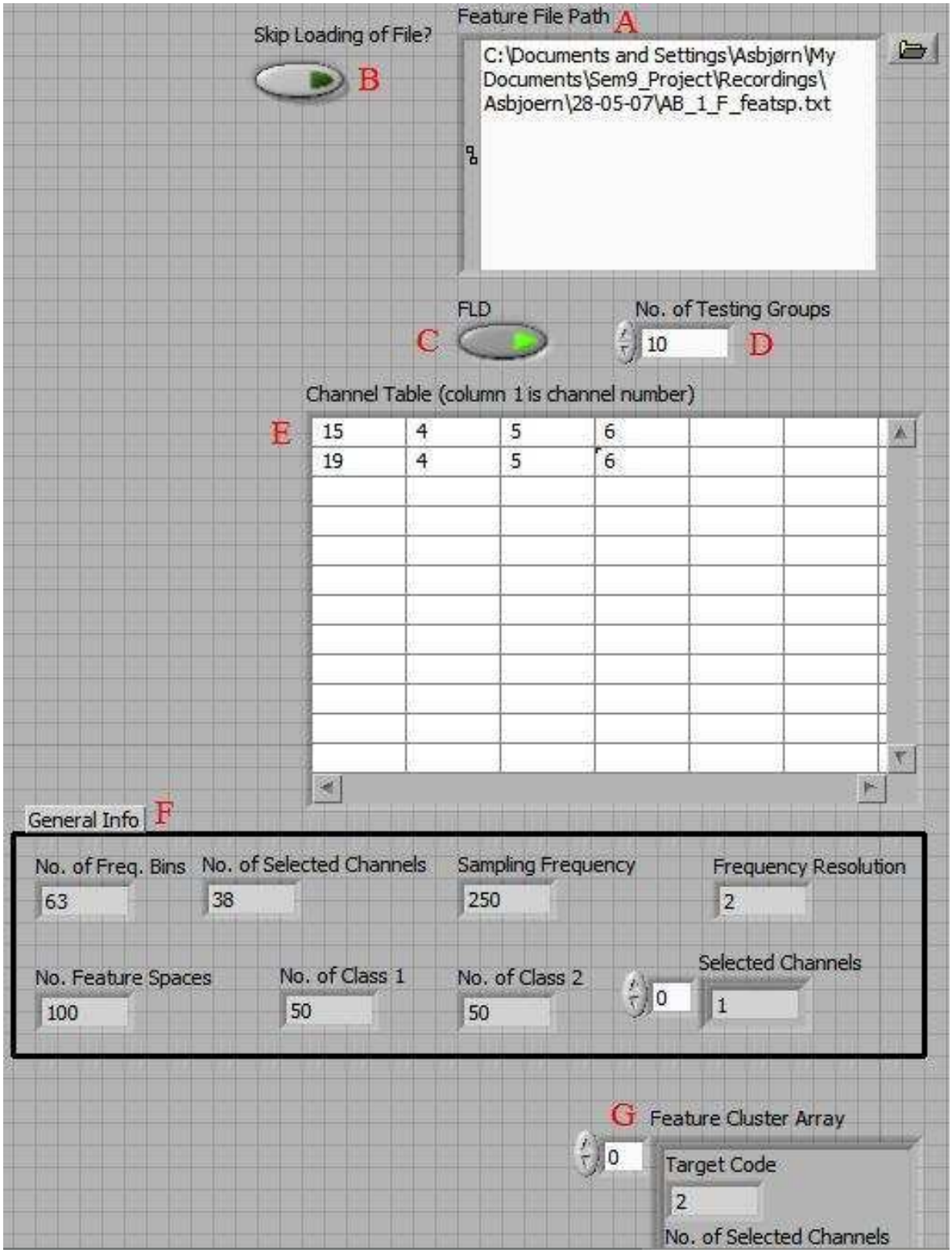

Figure 6.1: The general options.

- E: A feature selection table. Here you can manually type in the features that are desired for examination. It has multiple purposes which will be described later under each of the program modes. In the first column are the numbers for the channels typed in. This number refer to their index number in the feature space file (starting at index 0). In the following columns are the frequency bins for the channels typed in, again by their index number. It is not necessary to have the same number of frequency bins selected for each channel except when trying to run R square under the "One by one examination" page. Our selection is channel 15 (C3) and 19 (C4) and the frequency bins 4, 5 and 6 (frequency 8, 10 and 12).

- F: Info about the feature spaces loaded from the feature space file.

- G: Storage object for the feature spaces after they have been loaded.

6.2 One by One Examination

This program mode is used when the distinctiveness of the features are to be examined independently one by one. The whole feature space can be processed or only those specified by the Channel Table. One feature bin will be taken from all of the feature spaces and be random divided in to the number of groups specified by No. of Testing Groups. There will be the same number of feature bins in all groups. The feature bins are then tested through the method of Cross-Validation, see 3.5.2. The scores for successful classified lefts and right are being stored for every feature. At the same time is the R square values, see 3.5.1, being computed. If the feature spaces have been limited by selection in the Channel Table are the resulting indexes in reference to the index in the Channel Table and not the previous index.

Figure 6.2 shows the controls and displays on the page.

- H: Path to the folder containing the R_square_anu.m file and the files with the sub functions that file uses. The program uses a MATLAB script node to handle these files. Therefore MATLAB 6.5 or later needs to be installed on the computer for it to work. As the node works through ActiveX it will only work on Windows computers. Only the path to the folder should be specified, not the file it self.

- I: The result of the R square computations. First column specifies the channel index and the second the frequency bin. Third column is the R square value. The array is sorted after highest R square value.

- J: Progress indicators telling how many features have been processed already and when is the end.

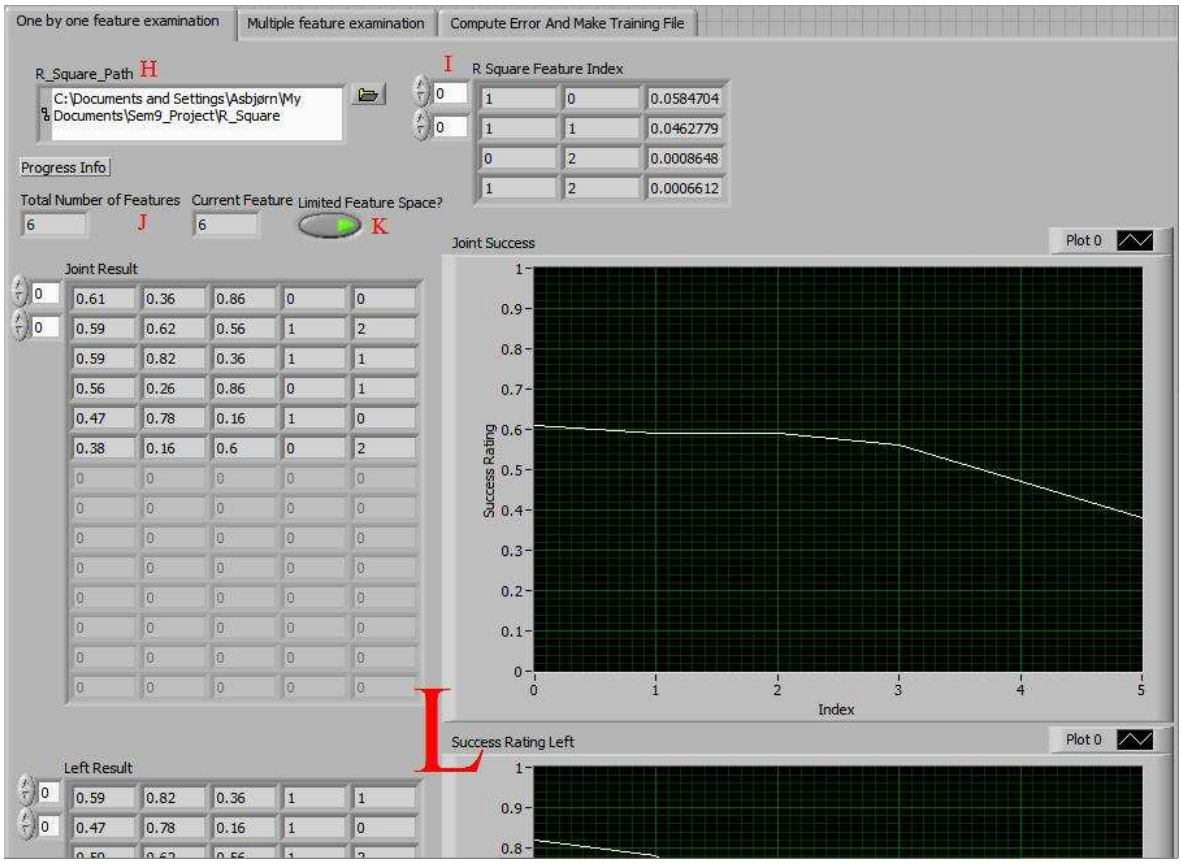

Figure 6.2: The "One by one examination" page.

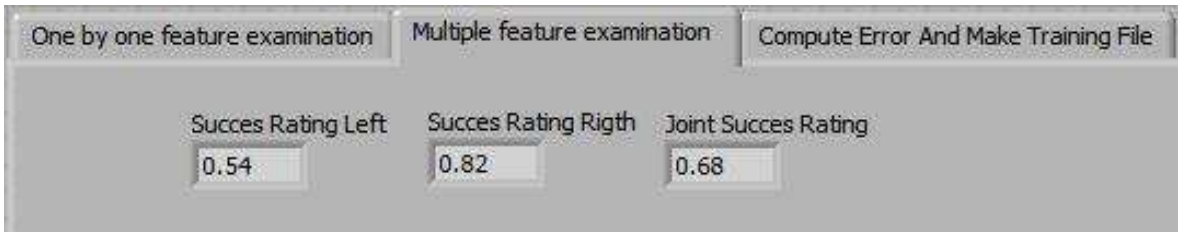

Figure 6.3: The "Multiple feature examination" page.

- K: Select if all of the feature space should be processed or only those specified by Channel Table.

- L: Scores for the features. First are the scores sorted by their Joint Success value. These values are then mean of success for left added the success for right. If a value has a high Joint Success it could indicate a high success for both left and right and would therefore be a good feature to separate the classes. Following the Joint Success is the scores again listed, but now sorted after best left and then best right. This is to see which single features are the best to classify either left or right.

6.3 Multiple Feature Examination

In this program mode is all of the feature bins selected by the Channel Table and used in a single test with Cross-Validation. The aim is to see how well the classification of the data is with multiple features. The scores are displayed on the page, figure 6.3. This is useful to compare the performance of a group of features that might have had good individual scores.

6.4 Compute Error and Make Training File

The program will in this mode generate the training file that the online system will use to train the classifier. Unlike the previous modes this one does not use Cross-Validation. Instead it computes one model for the selected feature bins over all the feature spaces. Like in the previous mode the feature bins are selected by the Channel Table. The program will also compute Best Left Error and Best Right Error and save them. See 3.4 for information about these values. The FLD values will be saved in the training file if FLD has been selected, otherwise only the Channel Table and the 'w'-values will be saved.

Figure 6.4 shows the page.

- M: Path and file name of the generated training data file. If the name of feature space file in process is of the pattern "name"_featsp.txt the name of the training file will automatically be generated to "name"_trainingdata.txt. The program will automatically

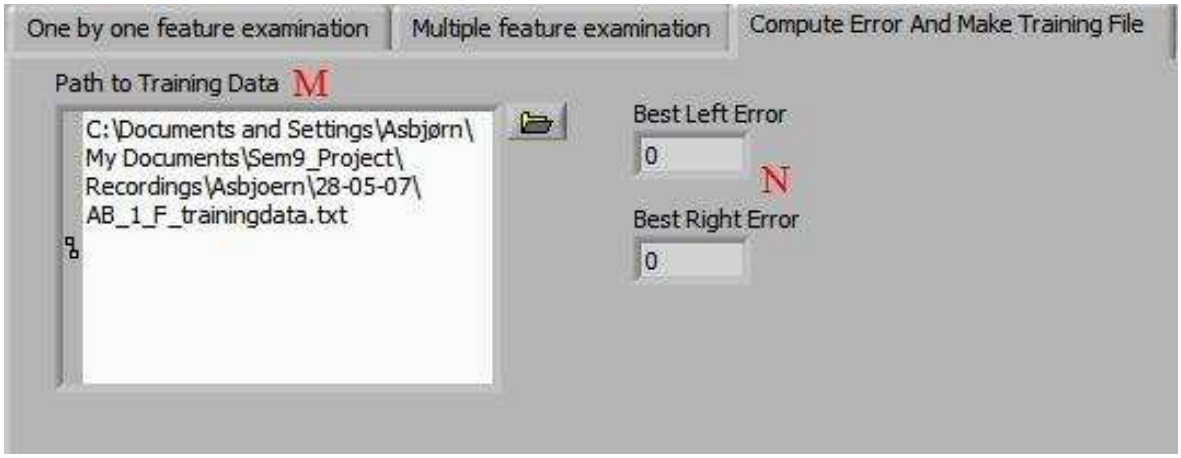

Figure 6.4: The "Compute error and make training file" page.

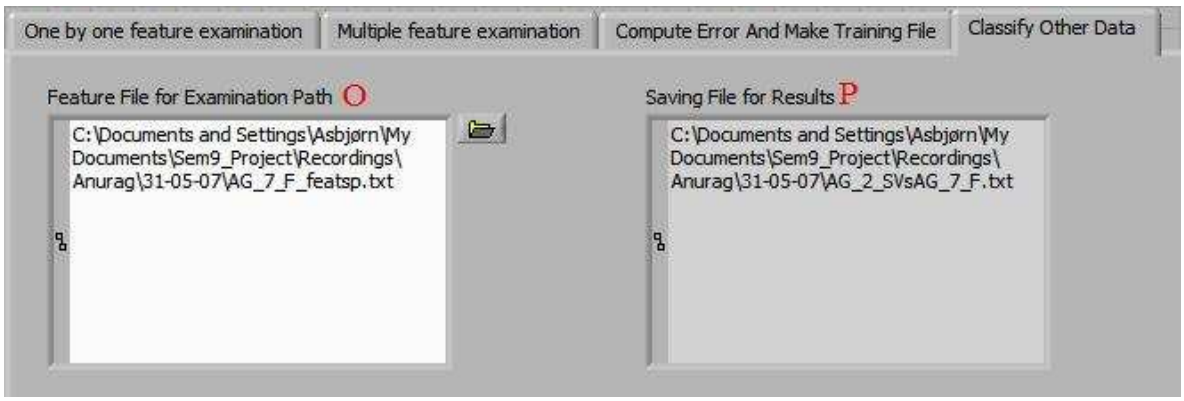

Figure 6.5: The "Classify Other Data" page.

overwrite any file of the same name without questioning the user.

- N: The computed values of Best Left and Best Right Error.

6.5 Classify Other Data

This mode is used when it is desired to train the classifier on one set of feature space and classify another set. The feature bins used to train and classify are selected by the Channel Table. The results will be saved in a file. The mode is useful when you want to examine offline if a different selection of features might have improved the classification. Figure 6.5 shows the page.

- O: Path and name of the feature space file wanted for examination.

- P: Path and name of the file where the result of the classification will be saved. The name will be automatically generated if both feature space files end on "name"_featsp.txt. The file will contain two columns: The target state of the feature space and the classified result.

Chapter 7

Results

Addressing to our hypothesis, this section will provide the results for the later discussion on whether the subject is been able to train over multiple days of sessions.

7.1 Overall Success Rate and Success Rate Per Second

In order to investigate the same, we plot the results for the subject's performance over sessions every day. The bar plot shows information regarding subject's Success Rating (Classification Percentage) for each session on a particular day (1 Session = 10 trails) and also represents how many of lefts and rights targets have been classified correctly during a trail. The number of correct classified lefts are marked with a red colour while the number of correct classified rights are marked with a green. The two results are stacked on top of each other to provide the overall Success Rate.

We have also looked into the subject's performance based on classification per second (called Success Rate per Second), meaning instead of having 1 result for a 10s trail and comparing it with target code, we look at the classification result for every second and compare this to the target code. The Success Rate per Second could give a better resolution of the performance. Hence, in order to see if there exists any relation between the overall Success Rate and Success Rate per Second we plot them together.

We will be presenting each subject individually for each day.

7.1.1 Subject 1

For Subject 1 we had made 3 days of recording with 6, 8 and 7 sessions recorded on each day respectively.

Looking at the figure: 7.1, it can be seen that subject has a better classification for left targets than right. For 5 out of 6 sessions, the overall Success Rate is ranging from 50% to 60%. However, going by the Success Rate per Second, it shows that the subject classification for each session is actually better than overall classification result. Also, the Success Rate per Second more or less follows same pattern as that of overall Success Rate.

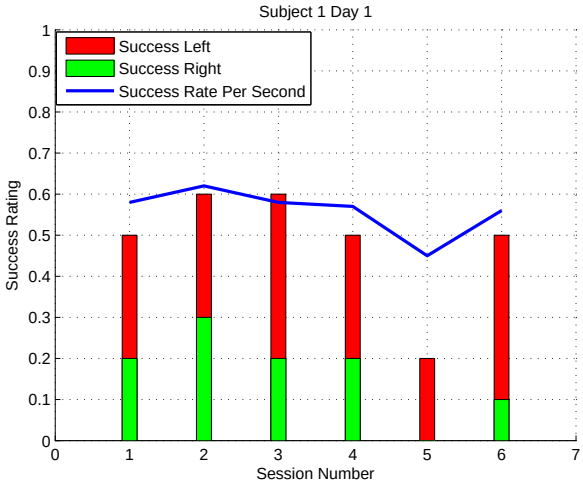

Figure 7.1: Subject 1 Performance on day 1.

Looking at the subjects performance on day 2 refer figure: 7.2, it can be observed that the overall Success Rate for day 2 is ranging from 40% to 80%. Success Rate per Second ranges only between 45% to 69%. But for most of sessions except 5,6,7, Success Rate per Second scores better than the overall Success Rate.

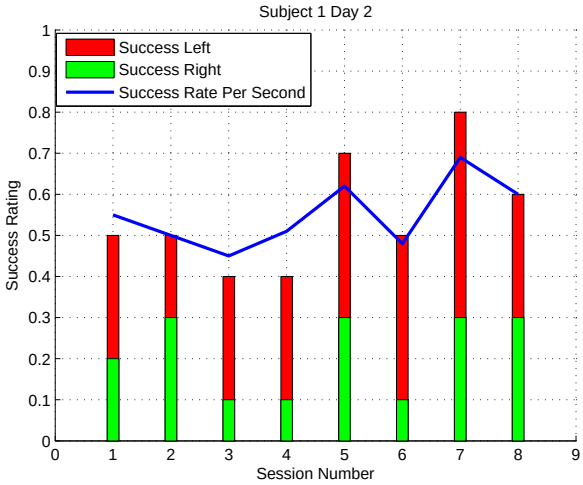

Figure 7.2: Subject 1 Performance on day 2.

For the 3rd day refer figure: 7.3, the overall Success Rate ranges from 50% to 80% and the

Success Rate per Second is from 55% t0 63%.

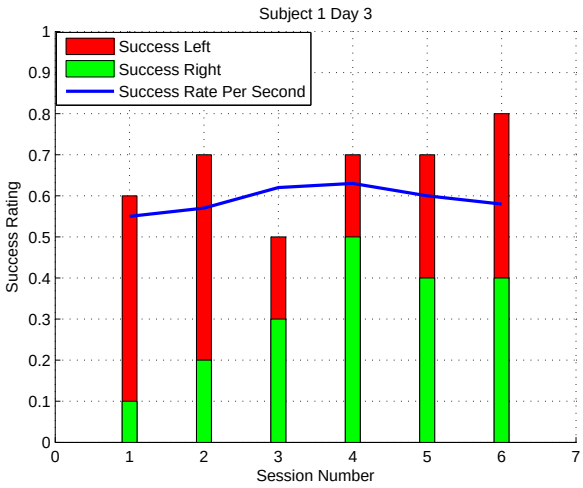

Figure 7.3: Subject 1 Performance on day 3.

7.1.2 Subject 2

For the subject 2 we made 4 days of recordings having 4, 6, 7, 7 sessions on the respective days.

From the figure: 7.4, it can be seen that subject has a flat performance of 50% for all the sessions during the day. The subject was not able to classify any right target during the day. The Success Rate per Second ranging from 45% to 53%, has given a classification of more than 50% for the session 2 while for the other sessions it was less than 50%.

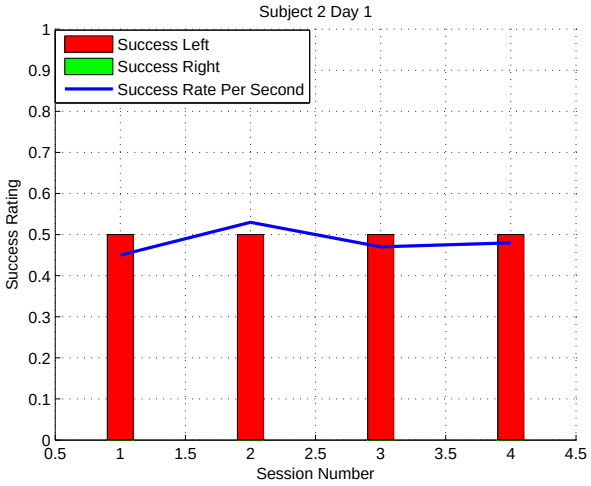

Figure 7.4: Subject 2 Performance on day 1.

On day 2 refer figure: 7.5, the overall Success Rate ranges from 50% to 60% whereas the Success Rate per Second ranges from 48% to 57%. During the last two sessions, subject was able to make rights and hence contributed to classification improvement.

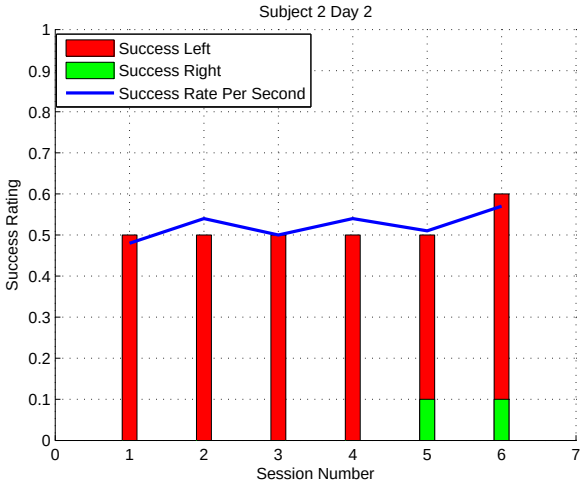

Figure 7.5: Subject 2 Performance on day 2.

Figure: 7.6 shows that during day 3 subject has a better overall Success Rate in comparison to previous days ranging from 50% to 70%. The Success Rate per Second has higher value than overall Success Rate for every session during the whole day. The Success Rate per Second range from 53% to 64%.

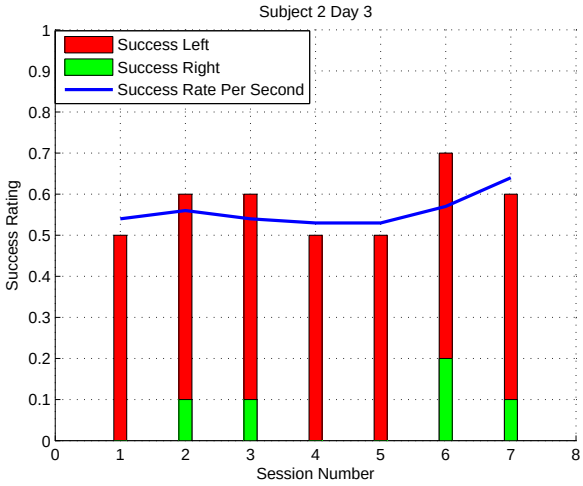

Figure 7.6: Subject 2 Performance on day 3.

For the 4th day, subject's overall classification rate rose to 80% for session 5 refer figure: 7.7. The overall Success Rate was ranging between 50% to 80% where as the Success Rate per

Second was between 44% to 60%.

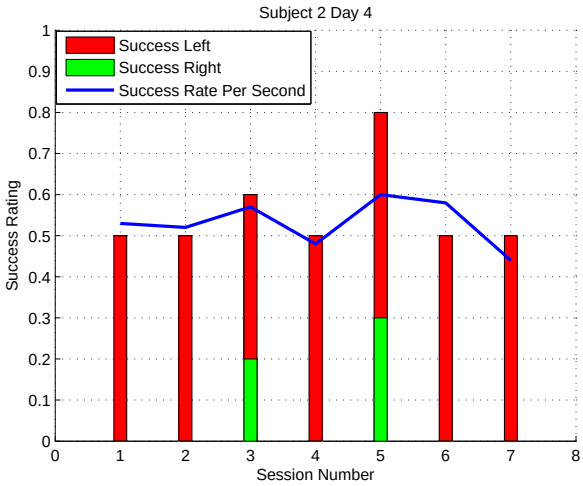

Figure 7.7: Subject 2 Performance on day 3.

7.2 Error Success Rate

The Error Success Rate is a value obtained by the classification based on g_{error}. In order to read about g_{error} please refer section: 3.4.g_{error} could be of interest to investigate as in case of classification by using the normal classification procedure there is a possibility to get a result state zero and not making any decision for either left or right. This happens if the numbers of classified lefts equals the number of classified right during a 10s trial. This would be highly unlikely in case of summation of the g_{error} values during the trial. Also, using g_{error} it is possible to define *"good left"* or *"bad left"* or in other words defining subject's relative performance to his best in regard to the screening data. The relative information g_{error} was given as feedback to the subject during the feedback sessions and he could use this to relate his performance to best possible. This is also why the summation is done on the normalised values for left and right. If the summation gives a positive value we mark that as a classification for left while a negative value is classification for right.

We are presenting the data for each subject on one particular day as an example for the performance of Error Success Rate.

Figure: 7.8 and figure: 7.9 shows the performance of subjects with Error Success Rate as well as the overall Success Rate. In case of subject 2, Error Success Rate is having approximately 10% greater classification rate in comparison to overall Success Rate except for session 3. For the sessions 2 and 7, the difference is almost equal to 20%. The Error Success Rate has shown a range of 50% to 80% in this case. And for the subject 1 also Error Success Rate has shown a better classification rate of 10% for 5 out of 8 sessions in comparison to overall Success Rate. For the session 1 Error Success Rate follows overall Success Rate but for the session 5 and 6

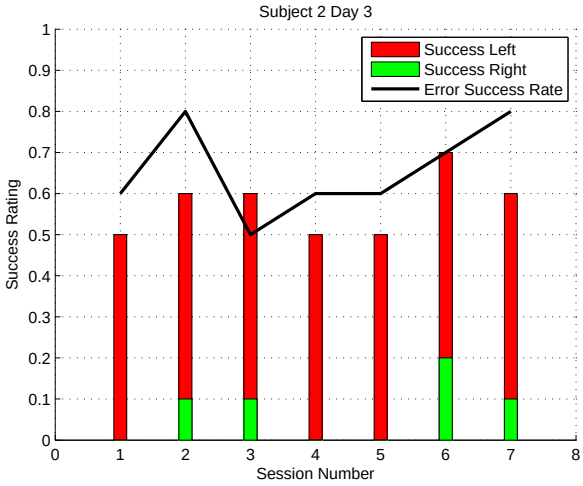

Figure 7.8: Comparison of overall Success Rate with Error Success Rate, Subject 2 Day 3.

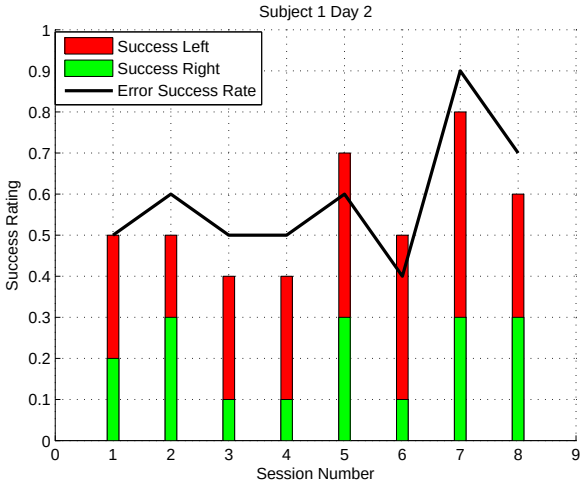

Figure 7.9: Comparison of overall Success Rate with Error Success Rate, Subject 1 Day 2.

it reduces by 10%.

7.3 Spectrum Plots using 6 Features Classification

In this section, we are presenting spectrum plots obtained by the autoregressive spectral method for the screening as well as the feedback sessions for the first 2 days averaged over all the sessions for each day. As the selected features for the classifier were at 8, 10 and 12 Hz from channel C3 and C4, we have plot the same features together with 6 and 14 Hz for both the channels in this plots. It is to be note that the frequency resolution for these plots is 2 Hz.

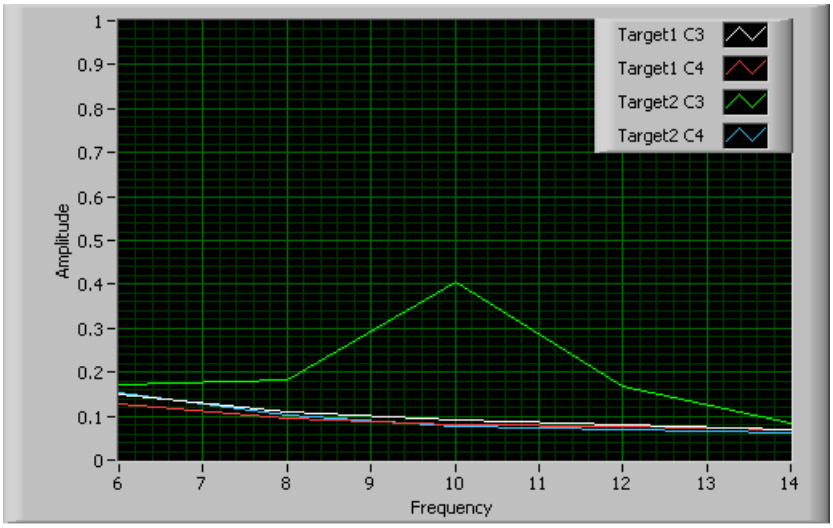

Figure 7.10: The power spectrum (microvolt) for the mu band (8-12 Hz), Subject 2 Screening session, Target1 = Target Left, Target2 = Target Right.

Figure: 7.10 shows the spectrum for C3 and C4 for the screening session. It can be observed that C3 channel shows a greater activity at 10 Hz for the target2 (target right). On the other hand for target 1 (target left) activity in both the channels at all the features seems to be same.

Observing the subjects performance for the 1st day, it can be seen that, there is no distinguished activity in any of channel for either target. And following subject's performance on 2nd day (figure: 7.12),it can be seen that 10 HZ feature at C4 for target 2 is more prominent, where as rest of all other features seems be behave in similar manner.

7.4 Comparison of Single Feature with 6 Feature Classification

In the previous section's plots, it was observed that there is a prominent activity in channel C4 at 10 Hz. So, we are interested in looking what is the classification rate if we would have used this single feature instead of 6. This process is done using the Offline Tools, please

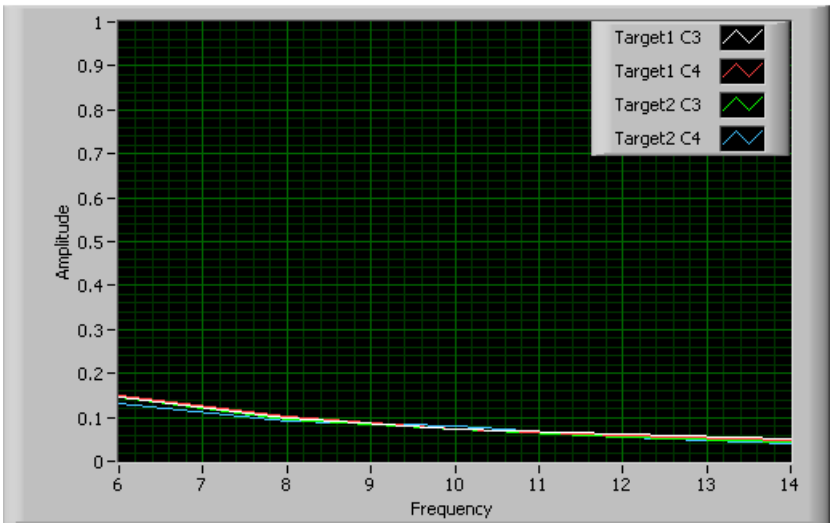

Figure 7.11: The power spectrum (microvolt) for the mu band (8-12 Hz), Subject 2 Day 1, Target1 = Target Left, Target2 = Target Right.

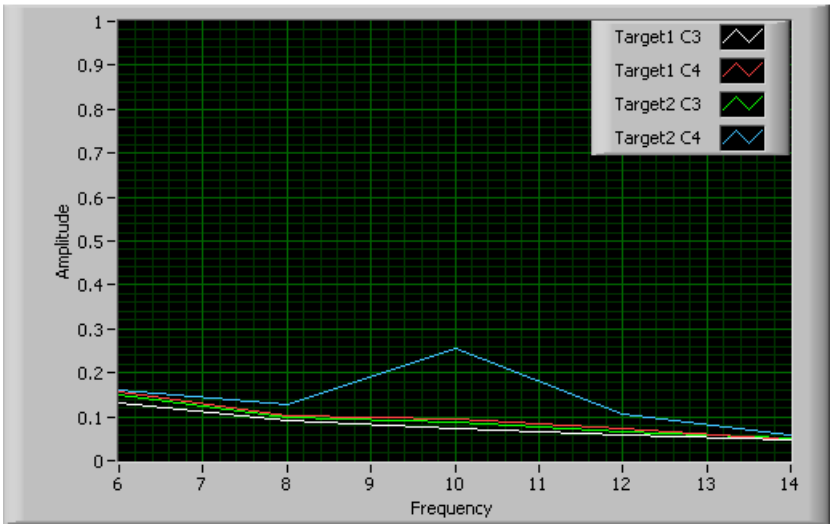

Figure 7.12: The power spectrum (microvolt) for the mu band (8-12 Hz), Subject 2 Day 2, Target1 = Target Left, Target2 = Target Right.

refer: 6.5.

We are presenting the performance of Subject 2 over the days during training. The bar plots plotted adjacent to each other are based on 6 features and single feature classification. They also present, how many of lefts and rights targets have been classified correctly during each session.

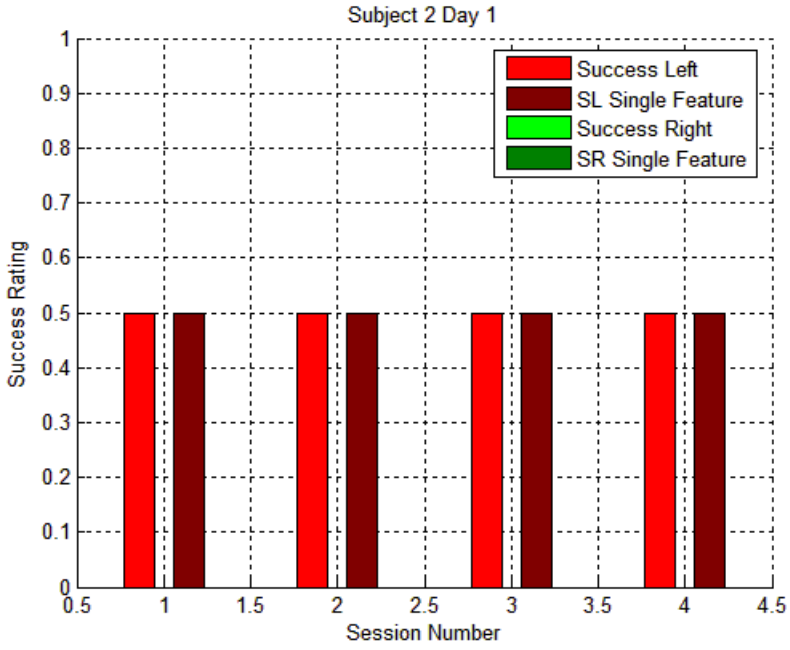

Figure 7.13: Comparison of 6 features classification with single feature classification, Subject 2 Day 1.

Figure: 7.13 shows the performance of subject for day 1. It can be seen that for both using 6 features and single feature, classification rate is 50% and there is not a single target right correctly classified. Where as on day 2 also (figure: 7.14), classification rate is similar in both case except for session 3 and 4 where there has been an addition of correctly classified right.

Following on to the performance of day 3 (figure: 7.15), the classification rate is identical for most of sessions except for session 5 and 6 where there is increment and decrement of 10% classification rate respectively. In both of these sessions, number of classified lefts is same for single and 6 features case but number of rights has shown variance.

Similar characteristics can be seen for day 4 (figure: 7.16), where both the classification methods yields same results except for session 3 and 7. In session 3 there is increment of 10% classification rate due to an additional classified right for single feature and in session 7 the classification rate falls by 10% due to an reduced classified left using single feature.

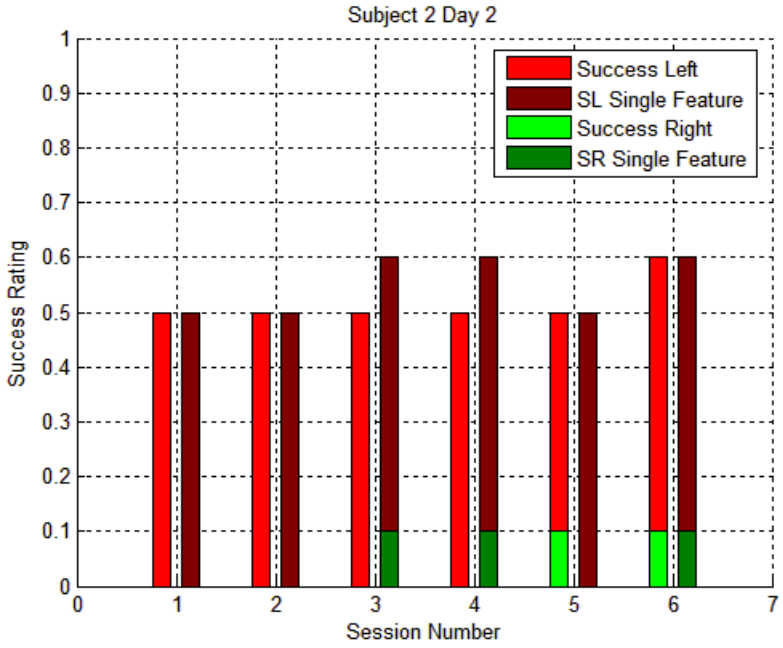

Figure 7.14: Comparison of 6 features classification with single feature classification, Subject 2 Day 2.

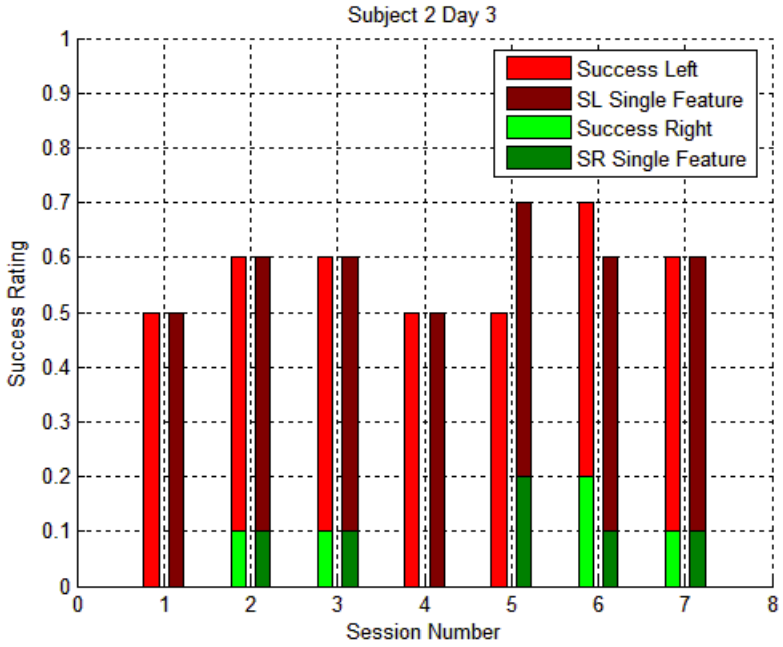

Figure 7.15: Comparison of 6 features classification with single feature classification, Subject 2 Day 3.

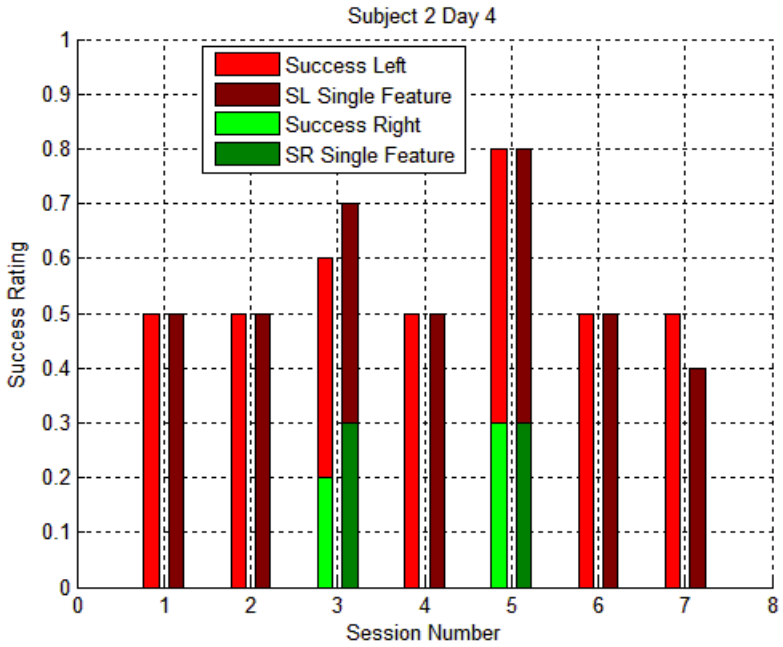

Figure 7.16: Comparison of 6 feature classification with single feature classification, Subject 2 Day 4.

Chapter 8

Discussion

As we only have two subjects and a limited amount of recorded data we look on our results as a case study.

8.1 The Biased Classifier

Before going deeper into our result we have to explain the concept called "the biased classifier". This is a classifier that have been fed with training data and have a preference to classify one state over another. A non-biased classifier would ideally classify an equal amounts of lefts and right when given random values as input. We evaluated the amount of bias in the Offline Tools using the Multiple feature examination before generating the training data. If the percentages between the numbers of successful classification of the screening data is too high ,i.e. the Success Rating Left could be 90% while Success Rating Right is 10%, training data generated on this screening session with the selected features would produce a heavily left biased classifier. The results of the Multiple feature examination for the screening data used to generate the training data with our selected features are given in decision tables 8.1 and 8.2. "Classified" is the decided state while "Target" is the true state.

Subject 1	Target Left	Target Right
Classified Left	47%	35%
Classified Right	53%	65%

Table 8.1: Multiple feature examination results for Subject 1.

Subject 2	Target Left	Target Right
Classified Left	83%	58%
Classified Right	17%	42%

Table 8.2: Multiple feature examination results for Subject 2.

The probability for correct classification for Subject 1 gets calculated to $P_C = 56\%$ and for

Subject 2 to $P_C = 63\%$. This shows that Subjects 2's classifier is performing slightly better than Subject 1's at least on the screening data. Still, Subject 2's classifier is also the one with the highest ratio between correct scored lefts and rights (nearly 2:1 ratio) so we call it a left biased classifier as it apparently classifies left much better than right. Subject 1's classifier only shows a slight bias toward right (1:1.4 ratio).

8.2 Training And Improvement

The average Success Rate per Session has been computed for both subjects over all days along with the standard deviation. The results can be seen in table 8.3.

Subject	Day 1	Day 2	Day 3	Day 4
1	48±15%	55±14%	67±10%	-
2	50±0%	52±4%	57±8%	56±11%

Table 8.3: Success Rate per Session based on overall Success Rate.

For both subjects there seem to be a steady increase in performance over the days. Compared to the success rate per session when using Success Rate per Second as the variable gives table 8.4.

Subject	Day 1	Day 2	Day 3	Day 4
1	56±6%	55±8%	59±3%	-
2	48±3%	52±3%	56±4%	53±6%

Table 8.4: Success Rate per Session based on Success Rate per Second.

Also here does it seem to be an increase, though not as much in the previous table. This indicates that it is possible to improve the overall Success Rate without improving much on the Success Rate per Second. This is possible if the subject equally distributes his correct classifications between trials which will make him score a much better overall result. An example would be a subject with a fixed Success per Second Rate of 60%. If he equally distributes this out between trials he will always score 6 out of 10 correct and thereby getting an overall Success Rate of 100%. As our feedback to the subject informs him how many correct classifications he is having during the trial it is possible he uses this to his advantage.

8.2.1 Performance of Subject 1

Even as Subject 1 had a right biased classifier it seemed through the sessions to be a bigger task for the subject to get a correct right classification than left. Already one day 1 was the majority of classifications for the left and not right. But it was also clear that it was possible for the subject to somehow influence classification to one side or the other as it did not seem

like some biased side was dominating. If the classification had randomly shifted between left and right we would have gotten more zero results for the trials and poorer overall Success Rate. On day 2 the subject steadily increased performance in both the overall Success Rate and in Success Rate per Second and this continued on day 3. While the value of the overall Success Rate and Success Rate per Second increased the standard deviation decreased, meaning that not only did performance increase it also become more consistent. It is not clear if the subject has reached his max performance level yet.

8.2.2 Performance of Subject 2

As mentioned earlier Subject 2 had a left biased classifier. This could explain why all trials during day 1 got classified as left even when the target was right. Though, the Success Rate per Second was fluctuating indicating that not all classifications was left. During day 2 the subject got a better hold of making rights as can be seen in the increase of Success Rate per Second. Also he managed to get two trials classified correctly as right by the end of the day. By day 3 the subject continued to improve the right classification and both the overall Success Rate and Success Rate per Second continued to improve. Day 4 did not show much improvement except for a single session with the top score for Subject 2 of 80% correct classification. Sadly, after this the subject lost focus and performed poorer in the last two sessions. For this subject the standard deviation increased over the days, following the increase of correct classifications of right. As it seems like the subject has no problems getting left classification it is expected that continued training will help to improve the right.

8.3 Error Success Rate

As mentioned about the point of interest in investigating Error Success Rate in section: 7.2.

In the case of no classification by overall Success Rate, it is still possible to decide on a classification for left or right using Error Rate. However, it could be argued that in order to avoid the no classification problem, the number of seconds in trail could be made 11 or 9 instead of 10 and this would have resulted in classification in all cases. But in that setup the classifier's biasing effect would influence the system, as for example if the 11th target code is same as that of biased side of classifier then it would potentially classify it based on the bias and not on the subjects performance. Hence, it will not be a good idea to use odd number of seconds in a trial to avoid the no classification problem. While in case of using Error Rate for the same trial we could be sure that we would be having a result even in the above mentioned cases.

From the table: 8.3 and table: 8.5 it can be seen that overall Success Rate and Error Success Rate does not follow same pattern in Success Rate per Session for Subject 1, which means that it will not be justified to use Error Rate to circumvent the problems of no classification as the classification result is generally widely different between the methods.

Subject	Day 1	Day 2	Day 3	Day 4
1	68±15%	59±16%	57±12%	-
2	43±1%	55±1%	66±11%	56±19%

Table 8.5: Success Rate per Session based on Error Rate.

The pattern in the Success Rate per Session is the same for Subject 2 though with even better performance, so the method can not be completely dismissed. It could be possible that if the Error Meter display was used instead of the Success Rating display as the primary feedback to the subject the Error Success Rate could improve.

8.4 Performance With a Single Feature

Because of the heavy left bias for Subject 2 we looked at the spectrum plot of the 6 features during the screening session and the feedback sessions. As seen in the results chapter the power is rather low for all 6 features and does not change much between the screening session for left and right. Except channel C4 at the feature bin 10 Hz during the recordings for target right. Looking at the average spectrum plots during the feedback sessions we noticed something odd. During day 1, where the subject go nothing but left classification in his trials, there were very little difference in the features. Only one day two and the following days would a feature show a difference. But not in C3 as during the screening but instead in C4 still at 10 Hz in the recordings for right. As we classify with a linear combination in the FLD a change in feature (C4, 10 Hz) would be similar to a change in (C3, 10 Hz) when there is very little variance between the rest of the features.

To see if the discriminant feature (C4, 10 Hz) potentially would have been a better classifier on its own we used the Offline Tools to compute the classification rate for all feedback sessions with the original screening data but now only using this single feature. Table 8.6 shows the success per session in the two cases.

Number of features	Day 1	Day 2	Day 3	Day 4
6	50±0%	52±4%	57±8%	56±11%
1	50±0%	55±5%	59±7%	56±14%

Table 8.6: Success Rate per Session based on a single feature (C4, 10 Hz) for Subject 2.

So by using this single feature we see slightly increased performance than by using the six. It can not be said that this feature would have been a good to have trained the classifier on after the screening session though. Most obviously was it not C4 but C3 that had the discriminating feature in the screening session.

8.5 Information Transfer Rate

Having some values for the performance of our system and knowing its speed it is possible for us to compute its information transfer rate. This is a widely used parameter indicating speed and accuracy of a BCI system. For further detail we refer to [6]. If we base our systems accuracy on the mean of the performance of our subjects on their last day we end up with 61%.

If we base the speed of our system on the decision of a trial (10 sec) we get a transfer rate of: 0.2 bit/min.
If the base the speed of our system on the decision rate of the per second classification we get a transfer rate of: 2.1 bit/min.

Compared to other BCI system with a transfer rate of 5-25 bit/min our system is very slow. Both due to the limited amount of targets and the low success rate. As we had no plans initially about making a speedy system we do not consider it a problem a the current point of development.

Chapter 9

Conclusion

To test our hypothesis, *Using data from a single screening session to train a classifier, a subject can improve his classification success rate over multiple sessions when using an online program that provides feedback of his performance during the sessions*, we designed an online BCI system. The system included data acquisition, signal processing, performance feedback and storage of all data and setup parameters. To support the online system we also made some offline tools for feature selection and generating training data for the online classification.

We used two subjects for our research, and recorded data for multiple feedback sessions over 3-4 days. The success scores for correct classified lefts and rights for the subject was examinated. Different criteria for correct classification was used. One being the classified state in regards to the target state on a one second basis. Another, and our primary criterion, was an accumulated result based on the per second classification over a trial (10 seconds). Lastely we inspected a parameter indicating a quantisation of the classification. As we only had few subjects and limited number of sessions we do not have enough data for any statistically conclusive results. Instead we looked at our two subjects as case studies.

As our results and discussion shows it was not only possible for the subjects to influence the outcome of the classifier but there were also a noticable increase in performance over the days. This was true when using our primary criterion as well as the per second based criterion for calculating the success rate. These results support our hypothesis.

To avoid the problems of no classification during trials with an even amount of seconds of classification and the influence by a biased classifier we examinated our last criterion. It did not show the same trends as the other two but as it was not intended as the primary feedback to the subject that does not conclude anything. What it does conclude, is that you can not change to this criterion in a case of no classification with the primary criterion as the results are quite different between the two.

Future Work

More subjects and sessions needs to be recorded to support or dismiss our hypothesis. Also it could be attempted to use different features for the classifier than the ones we had selected. Instead R square could be used to find the features with largest seperability between states.

Bibliography

[1] . [online, cited May, 2007]. Available from: `http://www.massgeneral.org/childhoodepilepsy/images/glossary/gt_cerebrum-l.jpg`.

[2] Marjan Jahanshahi, Mark Hallet. *The Bereitschaftpotetial-Movement Related Cortical Potentials.* Kluwen Academic and Plenum Publishers, 2003.

[3] Klaus-Robert Muller, Charles W. Anderson, and Gary E. Birch. Linear and Nonlinear Methods for Brain Computer Interfaces. *IEEE Transactions On Neural Systems And Rehabilitation Engineering*, 11-2:165, 2003.

[4] Webster. *Medical Instrumentation-Application and Design.* John Wiley and Sons, 1998.

[5] Theresa M. Vaughan, Jonathan R. Wolpaw, and Emanuel Donchin. EEG Based Communication: Prospects and Problems. *IEEE Transactions On Rehabilitation Engineering*, 4, 1996.

[6] Jonathan R.Wolpaw (Guest Editor), Niels Birbaumer,William J. Heetderks, Dennis J. McFarland, P. Hunter Peckham, Gerwin Schalk, Emanuel Donchin, Louis A. Quatrano, Charles J. Robinson, and Theresa M. Vaughan (Guest Editor). Brain Computer Interface: A Review of First International Meeting. *IEEE Transactions On Rehabilitation Engineering,*, 8, June,2000. number 2.

[7] Kalcher J, Flotzinger D, Neuper C, Golly S, Pfurtscheller G. Graz Brain Computer Interface II: towards communication between humans and computers based on online classification of three different EEG patterns. *Med Biol Eng Comput*, 34:382–388, 2005.

[8] Millan J del R, Mourino J, Babiloni F, Cincotti F, Varsta M, Heikkonen J. Local neural classifier for EEG based recognition of mental tasks. *Procedings of International Joint Conference on Neural Networks*, pages 632–636, July,2000.

[9] Birbaumer N, Ghanayim N, Hinterberger T, Iversen I, Kotchoubey B, Kubler A, Perelmouter J, Taub E, Flor H . A spelling device for paralysed nature. *Journal of Applied Physiology*, 398:297–298, 1999.

[10] Wolpaw JR, McFarland DJ. Multichannel EEG-based brain-computer communication. *Electroencephalography Clinical Neurophysiology*, 90:444–449, 1994.

[11] McFarland DJ, McCane LM, David SV, Wolpaw JR. Spatial filter selection for EEG based communication. *Electroencephalography Clinical Neurophysiology*, 103:386–394, 1997.

[12] G. Pfurtscheller, FH Lopes da Silva. Event Related EEG/MEG synchronisation and desynchronisation: basic principles. *Clinical Neurophysilogy*, 110:1842–1857, 1999.

[13] Pfurtscheller G, Flotzinger D, Mohl W, Peltoranta M. Prediction of the side of hand movements from single-trial multi-channel EEG data using neural networks. *Electroencephalogr Clin Neurophysiol.*, 82(4):313–315, 1992.

[14] Pfurtscheller G, J. Kalcher, C. Neuper, D. Flotzinger, M. Pregenzer. Online EEG classification during externally based hand movements using neural networks. *Electroencephalogr Clin Neurophysiol.*, 99:416–425, 1996.

[15] Gerwin Schalk, Member, IEEE, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R.Wolpaw. BCI2000: A General-Purpose Brain-Computer Interface (BCI) System. *IEEE Transactions On Biomedical Engineering*, 51:1034, 2004.

[16] BCI2000 Home Page [online]. Available from: `http://www.bci2000.org/BCI2000/Home.html`.

[17] Alvaro Fuentes Cabrera. A Phd Student Member at BCI Group At SMI. *Department of Health Science and Technology, Aalborg University, contact information: vhooraz@smi.auc.dk.*

[18] Eric Kandel, J H Schwartz, T M Jessell. *Principles of Neural Science*. McGraw Hill, 4th edition, 2000.

[19] G. Pfurtscheller and A. Aranibar. Event-Related Cortical Desynchronization Detected By Power Measurements Of Scalp EEG. *Electroencephalography and Clinical Neurophysiology*, 42:817–826, 1977.

[20] Decety J. The neurophysiological basis of motor imagery. *Behav. Brain Res.*, 77, 1996.

[21] Hallet M, Fieldman J, Cohen L.G, Sadato N, Pacula-Leone A. Involvement of primar motor cortex in motor imagery and mental practice. *Behav. Brain Science*, 17, 1994.

[22] Joseph Mourino, R Milan, F Cincotti. Spatial Filtering in the training process of the Brain Computer Interface. *23rd Annual EMBS International Conference*, 2001.

[23] C. Guger, H. Ramoser, and G. Pfurtscheller. Real Time EEG Analysis with Subject Specific Spatial Patterns for a Brain Computer Interface (BCI). *IEEE Transactions on Rehabilitation Engineering*, 8, Dec 2000.

[24] O Bertrand, F Perrin, J Pernier. A theoritical justification of the average reference in topographic evoked potential studies. *Electroencephalography and clinical Neurophysiology*, 62:462–464, 1985.

[25] Dave Meko. [online, cited March, 2007]. Available from: `http://www.ltrr.arizona.edu/~dmeko/notes_7.pdf#search=%22detrending%22`.

[26] Mark Polak, Alexsander Kostov. Feature Extraction In Development Of Brain-Computer Interface: A Case Study. *IEEE Engineering in Medicine and Biology Society*, 20, 1998.

[27] Gert Pfurtscheller, Christa Neuper, Alois Schogl, and Klaus Lugger. Separability of EEG Signals Recorded During Right and Left Motor ImageryUsing Adaptive Autoregressive Parameters. *IEEE Transactions On Rehabilitation Engineering*, 6:316, 1998.

[28] 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 Transactions On Rehabilitation Engineering*, 8:216–219, 2000.

[29] Nai Jen Huan and Ramaswamy Palaniappan. Neural network classification of autoregressive features from electroencephalogram signals for brain computer interface design. *Journal Of Neural Engineering*, 2004.

[30] J.M Spyers-Ashby, P.G Bain, S.G Roberts. A comparison of fast fourier transform (FFT) and autoregressive (AR)spectral estimation techniques for the analysis of tremor data. *Journal of Neuroscience Methods*, 83:35–43, 1998.

[31] P. J. Franaszczuk, K. J. Blinowska, and M. Kowalczyk. The Application of Parametric Multichannel Spectral Estimates in the Study of Electrical Brain Activity. *Biol. Cybern*, 51:239–247, 1985.

[32] Nai-Jen, Huan, Ramaswamy Palaniappan. Classification Of Mental Tasks Using Fixed And Adaptive Autoregressive Models Of EEg Signals. *IEEE*, 2004.

[33] Dan Mulally, Hyung W. Paik. Power Spectral Density Estimation Simulation using the Maximum Entropy Method.

[34] R Palaniappan, P.Raveendran, Shogo Nishida and Naoki Saiwaki. Autoregressive Spectral Analysis and Model Order Selection Criteria for EEG Signals.

[35] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley Interscience, 2nd edition, 2004.

[36] Deon Garrett, David A. Peterson, Charles W. Anderson, and Michael H. Thaut. Comparison of Linear, Nonlinear, and Feature Selection Methods for EEG Signal Classification. *IEEE Transactions On Neural Systems And Rehabilitation Engineering*, 11-2:141, 2003.

[37] Marco Mattiocco. Contact information: m.mattioccohsantalucia.it.

[38] Wenyan Jia, Xianghua Zhao, Hesheng Liu, Xiaorong Gao, Shangkai Gao, Fusheng Yang. Classification of Single Trial EEG during Motor Imagery based on ERD. *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, 2004.

[39] Ming Cheng, Wenyan Jia, Xiaorong Gao, Shangkai Gao, Fusheng Yang. Mu rhythm-based cursor control: an offline analysis. *Clinical Neurophysiology*, 115:745–751, 2004.

[40] S. J. Roberts, W.D. Penny. Real-time brain-computer interfacing: a preliminary study using Bayesian learning. *Med. Biol. Eng. Comput*, 38:56–61, 2000.

[41] Clemens Eder. A Phd Student Member at BCI Group At SMI. *Department of Health Science and Technology, Aalborg University.*

[42] Pfurtscheller G, Lopes da Silva. *Handbook of Electromyography and Clinical Neurophysiology-Event Related Desynchronization.* Elsivier, 1999a.

[43] Simon Haykin. *Adaptive Filter Theory.* Prentice Hall, 3rd edition, 1996.

Appendix

Appendix A

Study on the NetReader C++ Program

A.1 Introduction

This document describes our research on the NetReader program and its code. The NetReader program is used to view EEG either recorded or stored signals through a network connection. It was originally released together with the Scan4.3 program package from Neuroscan. It was later modified by Antanas Veiverys for usage in the SS-VEP online BCI study by Alvaro Cabrera. This modification involved some separation of functionality into a dynamic library link (dll) module and addition of some simple signal processing. As little information is available on the program dynamics this document is meant as general description of the NetReader program and the modifications our group provided to it. This should make it easier for people with further interest in modifying the program. The program is written in C++ using Microsoft Foundation Classes (MFC) as framework to be able to use a Windows interface.

A.2 Program Functionality

A.2.1 Data Transmission

The NetReader program is a client program which runs in conjunction with the Acquire data acquisition program. This program is also a part of the Scan4.3 program package. Acquire acts as server feeding NetReader with samples through a TCP protocol. The data is sent in specific packages. Further information about these packages can be found in the documentation for the Acquire program, which is copied in this document in section A.5.1. This gives the possibility of running NetReader on a different computer than Acquire though both programs also can also run on the same. The setup is seen in figure A.1.

The Acquire program can run either in reording or simulation mode. To initialize the simulation mode you need to setup the default parameters for the amplifier. Press the "Edit"-tap and select the "Overall Parameters" option. From the new window select the "Amplifiers"-tap

Figure A.1: System setup

and press "Ok" to the two message boxes that will appear. Then close the "Setup"-window by pressing "Ok". You can now start simulation by pressing the "Start Data Pisplay"-button.

A.2.2 Primary Control Window

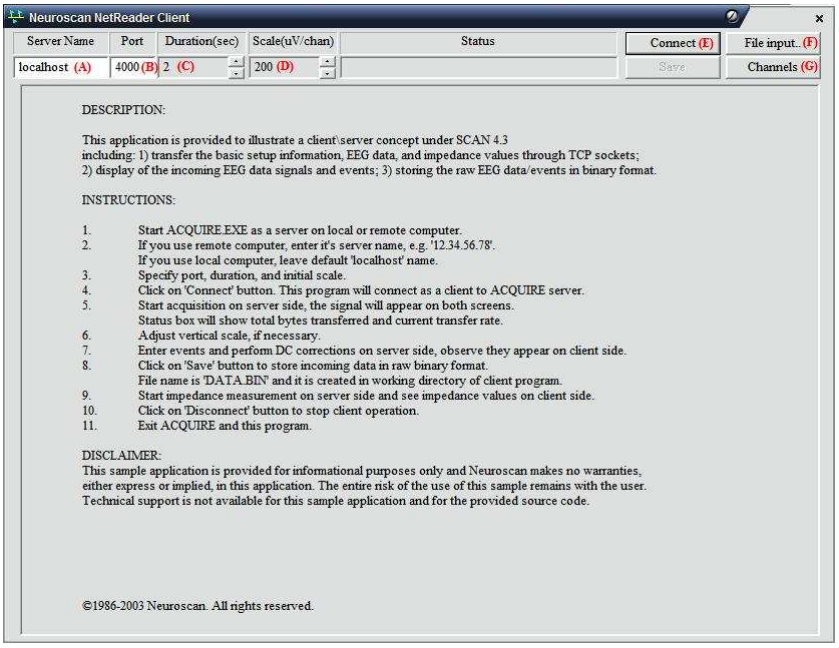

Figure A.2: Primary Control Window

Before the data exchange between Acquire and NetReader can happen they need to be setup to be able to find each other. This is firstly done by specifying the machine name or host where Acquire will be running. If both programs are running on the same machine this would be 'localhost' which has to by be typed in the 'Server Name' field (A). If Acquire is running on a different computer the IP-address of that computer has to go to that field. Then the port number has to be specified which of course should be the same for Acquire and NetReader. This value goes to the 'Port' field (B). The functions of fields C and D were not really tested but at least they got nothing to do with the server/client setup. When the server is running in Acquire press 'Connect' (E) to start the client. After starting the recordings in the Acquire program the signals will also be displayed in this window.

The 'File Input..' option (F) was made by Antanas Veiverys and allows you to load an .asc

file into the program. This file is a stored recording. You have to setup channels through the 'Channels' option (G) to have any process on these stored recordings going on. Also, the 'Connect' button has to be pressed once no matter if you want to connect to a server or not before you can load .asc files. The 'Channels' option opens the Channel Selection window.

A.2.3 Channel Selection Window

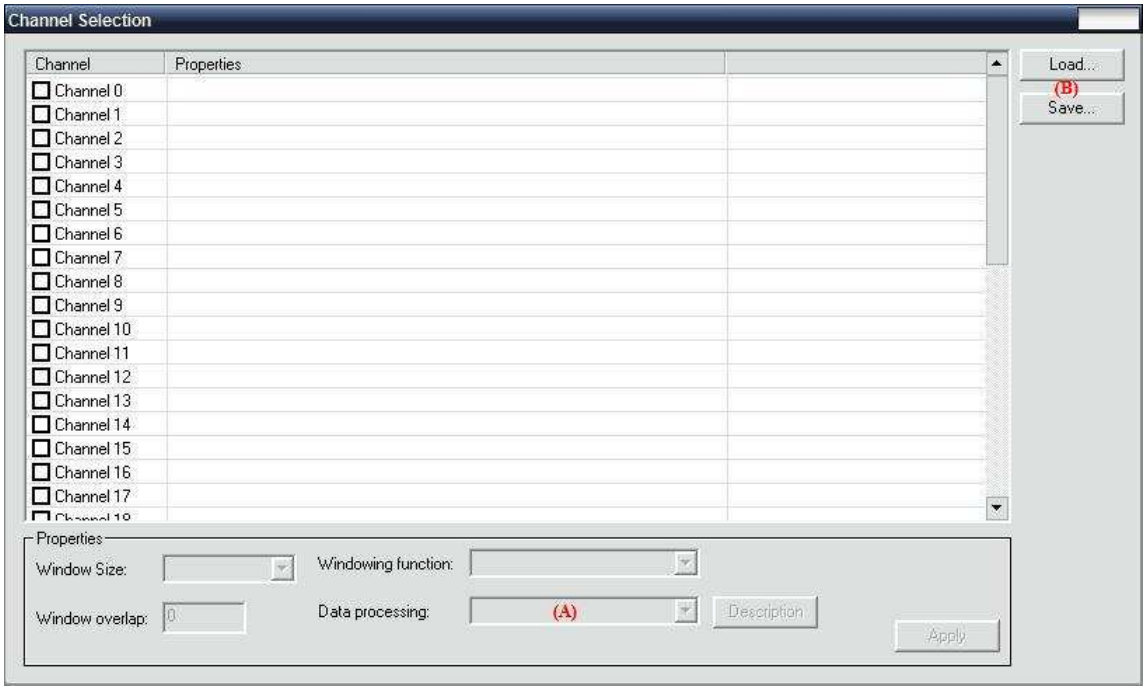

Figure A.3: Channel Selection Window

This window controls the setup for the channels you are receiving. The window list 40 channels listed from 0 to 39 which put some limitations on the channel setup you can use in the Acquire program. For each channel you can select a data process operation (A). This means that the data process on an individual channel is independent of the process on other channels and that you can not join channels together for comparison. The option 'Save' (B) allows you to save a channel setup, and option 'Load' allows you to load one. The data processes available are Discard data, File output and Build-in FFT.

The discard data operation is a bit redundant as all channels which are not marked for anything else do this by default.

A.3 Dynamic Library Links

Dynamic Library Links (dll) are used to provide code functionality to programs without revealing the code of this functionality. To use a dll you need to know the name and syntax of the functions it contains before you can use them. After having build your program the dll is required to be accessible to the executable file (normally the .exe-file) for the functions to be available. More than one program can use functions from a dll at a time.

Antanas Veiverys changed the structure of the NetReader and placed the TCP client interface functionality in a dll on its own using MFC support. This means that the dll does not export the syntax of its functions so you can not load them in a traditional C-manner. Instead you have to include the acquire.h file in your code and link with the acquire.lib (library file) to build the program. You still need to have the acquire.dll available to the executable file during run-time.

A.4 Modularization of the NetReader Program

Our primary modification to the NetReader program was to take the signal processing possibilities used in the Channel Selection Window and also place them in .dll-files. These also ended using MFC support. This lead to 3 dll's:

- DiscardPluginDll.dll, which discards the data.
- FilePluginDll.dll, which saves data.
- FFTPluginDll.dll, which does the fft operation and frequency recognition.

By moving these functionalities into their own dll's it gives the possibility of using them later on in other applications. The interface for these 3 dll's are similar.

- CString Name();
  This function returns the name of the primary function in the dll.
- CString Description();
  This function returns a short description of the primary function.
- int Execute(float *data, const int fs, const CChannelSettings *chan);
  This function executes the primary function of the dll. The input is a pointer to the data, the sampling frequency and the channel setup. The return value is 0 for success got the FilePluginDll.dll
- int Destroy();
  This function clears the memory usage of the dll.

It should be noted that the FFTPluginDll.dll does FFT by using the code provided by Laurent de Soras in the files FFTReal.h and FFTReal.cpp so if the dll is to be recompiled and build it needs access to these.

A.4.1 Program Flowchart

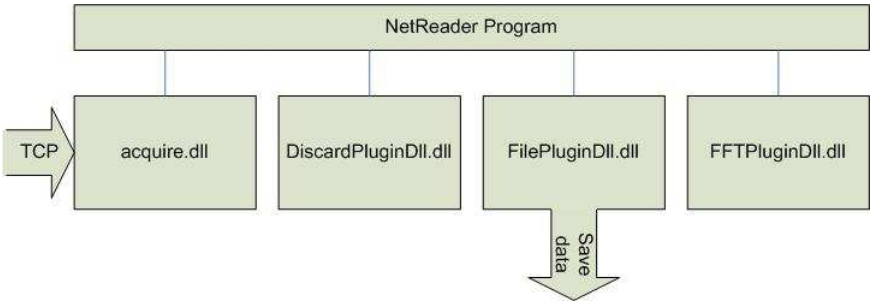

Figure A.4: Program flowchart

A UML diagram of the NetReader program can be found in the supplementary section A.5.2.

A.5 Supplementary

A.5.1 Acquire TCP Packages

This text is copied from the Acquire documentation regarding the TCP.

Packet structures and command set

Who should read this section? This section is intended for users who want to write their own client program to pull data online from the ACQUIRE server during acquisition. The ACQUIRE program uses stream sockets (neither datagram nor raw sockets) to guarantee data delivery in the proper order. Refer to Windows Socket Specification or books on TCP for more information on this socket type and socket programming.

Packet structure. All information sent between the client and server is organized as packets. Each packet consists of a 12-byte header and an optional body. The header structure is the following:

```
typedef struct{
char m_chId[4]; // ID string, no trailing
WORD m_wCode; // Code
WORD m_wRequest; // Request
DWORD m_dwSize; // Body size (in bytes)
}PacketHeader;
```

If **body size** field is 0, then no body is sent. Otherwise, the receiving socket should read the specified number of bytes and process them according to meaning of the header's ID string, Code and Request fields.

Byte ordering. Different machine architectures store data using different byte orders. For example, Intel-based machines store data in the reverse order of Macintosh (Motorola) machines. Intel's byte order, called "little-Endian," is also the reverse of the network standard "big-Endian" order. The following table A.1 explains these terms:

Byte ordering	Meaning
Big-Endian	The most significant byte is on the left end of a word.
Little-Endian	The most significant byte is on the right end of a word.

Table A.1: Endian Explanation

ACQUIRE always sends the packet header translated to "big-Endian"(network) notation. But the content of the data and file packets (as proprietary information) is always sent in native Intel "little-Endian" format. This means that ACQUIRE performs the following operations on packet's header:

```
void PacketHeader::Convert(BOOL bSending)
{
if (bSending) {
// Convert from host byte order to network byte order
// (Little-Endian to Big-Endian) while sending
m_wCode = htons (m_wCode);
m_wRequest = htons (m_wRequest);
m_dwSize = htonl (m_dwSize);
} else {
// Convert from network byte order to host byte order
// (Big-Endian to Little-Endian) while receiving
m_wCode = ntohs (m_wCode);
m_wRequest = ntohs (m_wRequest);
m_dwSize = ntohl (m_dwSize);
}
}
```

Command set. The Header ID string consists of 4 upper case letters, without the trailing. It can be one of the following:

"CTRL" - control code
"FILE" - file packet
"DATA" - data packet

This ID specifies what kind of information is sent in other fields and in the optional body. The following table summarizes the possible combinations: figure A.5

Packet ID string	Code Value	Code Meaning	Request Value	Request Description	Body size
"CTRL"	1	General Control Code	1	Request for Version	0
			2	Closing Up Connection	0
	2	Server Control Code	1	Start Acquisition	0
			2	Stop Acquisition	0
			3	Start Impedance	0
			4	Change Setup	0
			5	DC Correction	0
	3	Client Control Code	1	Request for EDF Header	0
			2	Request for AST Setup File	0
			3	Request to Start Sending Data	0
			4	Request to Stop Sending Data	0
"FILE"	1	Setup file	1	Neuroscan AST Format	$\neq 0$
"DATA"	1	Information	1	Version Information	$\neq 0$
			2	Standard EDF Header	$\neq 0$
	2	EEG data	1	Neuroscan 16-bit Raw Data	$\neq 0$
			2	Neuroscan 32-bit Raw Data	$\neq 0$

Figure A.5: Package Combinations

The Server computer sends general and server control codes, file and data packets. The Client side should send only general and client control codes, and be able to process (or simply drop) incoming file and data packets.

The formats for "Version Information", "EDF Header" and "Neuroscan 16-bit Raw Data" packets are presented in "acq_client.c" source code, included in the "NetReader" directory.

Important note. As ACQUIRE uses some other codes and requests internally, avoid the usage of numbers not presented in the table.

Program examples

Standard client operations. Normally, the client program first connects to the ACQUIRE server; second, it requests for configuration information and processes it; third, it asks for EEG data and then receives them for further processing/display/ storage; and, finally, it disconnects from the ACQUIRE server. The following are basic client actions:

1. Connect to ACQUIRE server through TCP socket.
2. Send "Request for EDF Header".
3. Read "DATA" packet with EDF Header and process it to extract electrode information.
4. Send "Request to Start Sending Data".
5. Read incoming "DATA" packets with 16/32-bit EEG data with further processing, displaying and storage.
6. Send "Request to Stop Sending Data".
7. Send "Closing Up Connection".
8. Disconnect from ACQUIRE server by closing up the client TCP socket.

See example code "acq_client.c" for such client implementation.

"Minimum" client operations. The basic sequence can be reduced. If you manually set the number of channels on the client side, then you may skip steps 2 and 3. Exiting the client program will also close all sockets, so you may also trim out steps 6-8. Finally, the minimum sequence of operations is:

1. Connect to ACQUIRE server through TCP socket.
2. Send "Request to Start Sending Data".
3. Read incoming "DATA" packets with 16/32 EEG data, process and display them.
4. Exit from client program (to automatically disconnect from ACQUIRE server).

See the sample code "acq_client_min.c" for implementation of the "minimum" client.

The "NetReader" directory contains the following program examples:

Name of file or project	Description	Language	Location (relative to SCAN4.3)	Platforms
acq_client_min.c	Minimum code for console ACQUIRE client	C	NetReader	Windows UNIX
Acq_client.c	Console-type ACQUIRE client with storage data to EDF file format	C	NetReader	Windows UNIX
AcqClient	Dialog-based ACQUIRE client with some graphical output	C++ MFC	NetReader\AcqClient	Windows
JavaClient	Console-type ACQUIRE client with storage data to file	Java	NetReader\JavaClient	Windows UNIX

Figure A.6: Supported Languages

Example "Reader" program and source code. Included with the 4.3 installation is a sample "reader" program that can be used to display the data on a second computer (without the need for a second SCAN license). To run the program, go to the \Scan4.3 folder and run the NetReader.exe program. You can run the program on the same computer where you have SCAN, if you wish. Follow the directions on the displayed screen. You should see the EEG signals on the "reader" at the same time as you see them in ACQUIRE. Event information will also be displayed. The source code for the program in found in the \Scan4.3\NetAcquire folder.

A.5.2 UML Diagram

This diagram was made by Antanas Veiverys before our changes to the NetReader program.

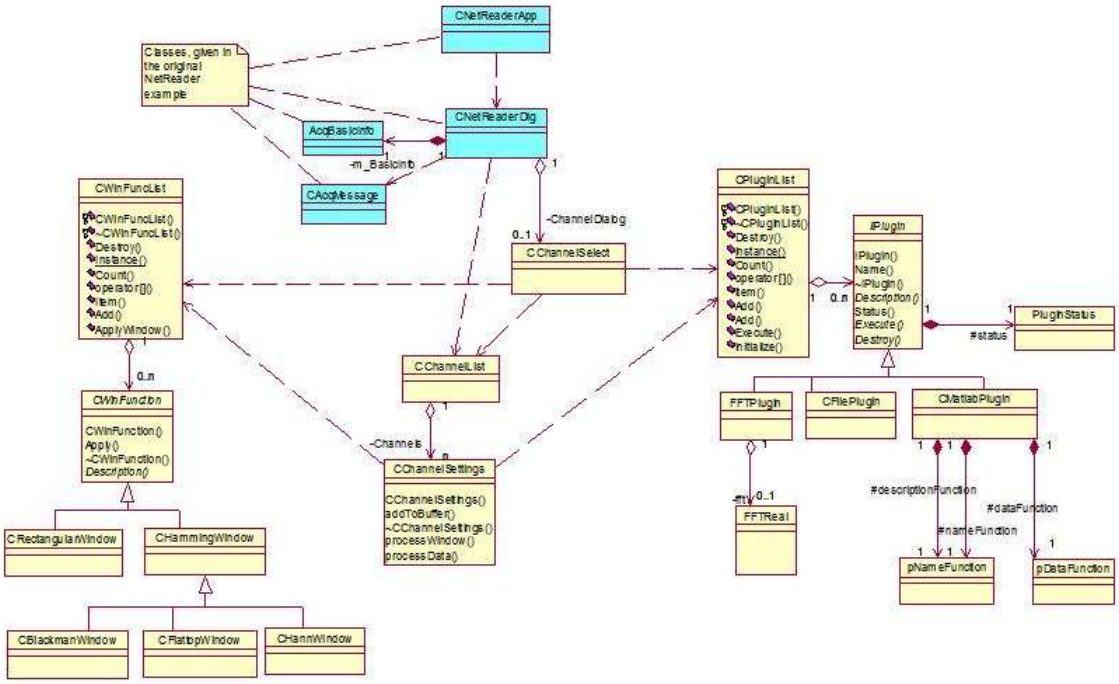

Figure A.7: UML Diagram

Appendix B

EEG Rhythms

The rhythms of the EEG are classified into various subtypes on the basis of the frequency and the associated cortical and the mental areas. The various rhythms shown in the figure: B.1 are :

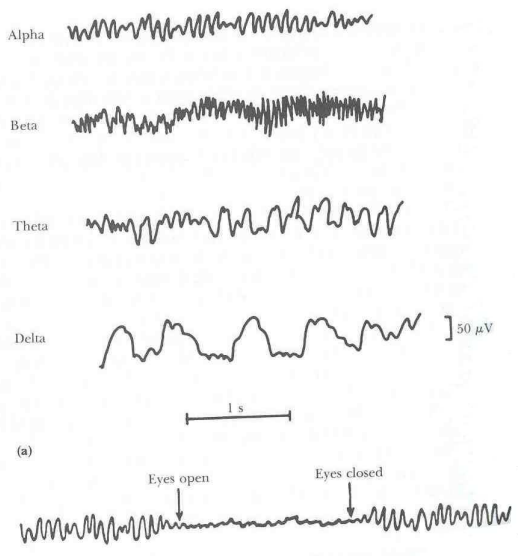

Figure B.1: Different types of normal EEG rhythms[4].

Delta

Delta rhythms falls in the category of waves having a frequency less than 3.5Hz. They occur for around 2-3secs. They are usually present during a deep sleep and also in new born babies.

Theta

Theta rhythms lies in the range of 4-7Hz. It is found typically in parietal and temporal region of children. However, it can also be seen in the adults experiencing emotional stress or disappointment.

Alpha

Alpha rhythms lies in the range of 8-13Hz and amplitude varying between 20-200μV. It has larger amplitude in comparison to other rhythms and more like sinusoidal in shape. Alpha activity can be recorded from occipital,parietal and posterior temporal regions of the cortex. It is associated with the wakefulness, physical relaxation and relatively mental inactivity.

Beta

Beta rhythms lies in the frequency range of 14Hz to 80Hz. The activity is usually asynchronous and its amplitude is always below 30μV. It can be divided into categories as lower beta activity and higher beta activity. The lower beta activity lies in the range 14-30Hz and is present in the frontal and central regions of the cerebral cortex. It signifies the limited mental activity and they disappears when the mental activity is enhanced. The higher beta activity is known as Gamma activity and is described below.

Gamma

Gamma rhythms lies in the range of 30-80Hz where 30-60Hz lies in lower band and 60-80Hz in higher band , they however similar in shape and location to lower beta rhythms but their amplitude is much lower to beta rhythms and is usually in μV. Gamma activity is associated with the high mental activity and is considered due to the processing of sensory,cognitive or sensorimotor information, for example planning a motor action.

The above section is based on the information provided in [4] and [42].

Appendix C

Signal Processing Tools Theory

C.1 AutoRegressive Models

A time series u(n),u(n-1),.....,u(n-M) represents the realisation of an autoregressive process (AR) of order M if it satisfies the difference equation:

$$u(n) + a_1^* u(n-1) + ... + a_M^* u(n-M) = v(n) \tag{C.1}$$

where $a_1, a_2, ..., a_M$ are constants called AR parameters and v(n) is a white noise process. The term $a_k^* u(n-k)$ is the scalar version of inner product of a_k and u(n-k), where k= 1,2,...,M.

The eq: C.1 can be rewritten as:

$$u(n) = w_1^* u(n-1) + ... + w_M^* u(n-M) + v(n) \tag{C.2}$$

where $w_k = -a_k$. It can be seen that the present value of the process u(n) is a linear combination of the past values of the process, u(n-1),.....,u(n-M), and a noise term. This equation can be represent as linear model:

$$y = \sum_{k=1}^{M} w_k^* x(k) + v \tag{C.3}$$

where y is said to be regressed on the $x_1, x_2, ..., x_M$ and since in eq: C.2 u(n) is regressed on past values of itself, it is called as *autoregressive* process.

The eq: C.1 can be represented as convolution of u(n) and a_n^* :

$$\sum_{k=0}^{M} a_k^* u(n-k) = v(n) \tag{C.4}$$

where a_0=1. By taking z-transform, it can be represented as:

$$H_A(z)U(z) = V(z) \tag{C.5}$$

where,

$$H_A(z) = \sum_{n=0}^{M} a_n^* z^{-n} \quad , \quad U(z) = \sum_{n=0}^{\infty} u(n) z^{-n} \tag{C.6}$$

With the white noise as input, we can produce the AR process u(n) as output as shown in figure: C.1, and the transfer function for the this process generator is equal to:

$$H_G(z) = \frac{U(z)}{V(z)} \tag{C.7}$$

From eq: C.5, it can be seen as:

$$H_G(z) = \frac{1}{H_A(z)} = \frac{1}{\sum_{n-0}^{M} a_n^* z^{-n}} \tag{C.8}$$

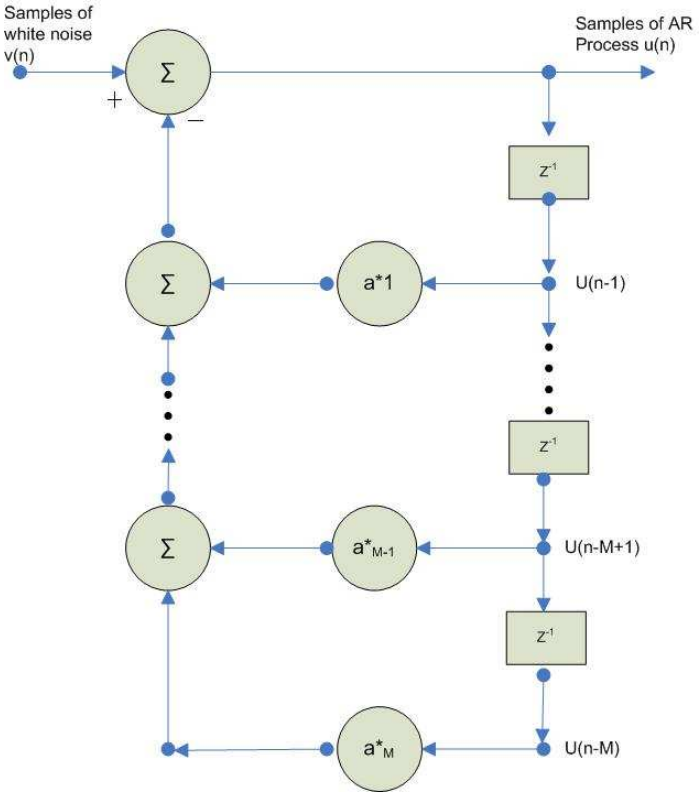

Figure C.1: AR Process generator.

The process generator shown in figure: C.1 is an all-pole filter. It is so called because $H_G(z)$ is completely defined by specifying the location of its poles:

$$H_G(z) = \frac{1}{(1 - p_1 z^{-1})(1 - p_2 z^{-1}).....(1 - p_M z^{-1})} \tag{C.9}$$

The parameters $p_1, p_2, ..., p_M$ are poles of $H_G(z)$; they are defined by the roots of the equation:

$$1 + a_1^* z^{-1} + a_2^* z^{-2} + ... + a_n^* z^{-n} = 0 \tag{C.10}$$

For the all-pole AR process generator to be stable, the roots of the eq: C.10 must lie inside the unit circle in the z-plane. This is also a necessary and sufficient condition for wide sense stationary of the AR process produced by figure: C.1.

C.2 Levinson-Durbin Algorithm

This section briefly describes the stages of Levinson-Durbin Algorithm. For more details please refer [43].

Initialize the algorithm by setting:

$$a_{0,0} = 1; \quad P_0 = r(0); \tag{C.11}$$

For m = 1, 2, ..., M, where M is the selected model order, compute:
Reflection Coefficients:

$$k_m = -\frac{1}{P_{m-1}} \sum_{i=0}^{M-1} r(i-m) a_{m-1,i} \tag{C.12}$$

AR Coefficients:

$$a_{m,i} = \begin{cases} 1 & \text{for } i = 0 \\ a_{m-1,i} + k_m a_{m-1,m-i}^* & \text{for } i = 1,2,....\text{m-1} \\ k_m & \text{for } i = \text{m} \end{cases}$$

Updating Error:

$$P_m = P_{m-1}(1 - |k_m|^2) \tag{C.13}$$

Appendix D

Testing of Modules and System

D.1 Selection of AR Model

D.1.1 Testing with sample data

An EEG data which were acquired during a study at Graz University of Technology, Austria by Gert Pfurtscheller and Alois Schlogl were used to test the order of system. Data were collected during a feedback session where the task was to control a feedback bar by means of imagery left or right hand movements. The Data was collected for 140 training and 140 test trials, at 3 channels, 128Hz sampling rate, 0.5-30Hz filter, each trial of 9s long.

As mentioned in chapter: 3, we chose Burg's method to calculate AR coefficients and observed the drop in FPE with order.

The mean FPE was calculated over all the 140 trails and for each order. Figure: D.1 shows FPE vs Model Order for all the 3 channels recorded and it can be seen that FPE falls till the order the 12 and then becomes almost consistent for channel 1 and 3, where as for channel 2 also the error seems to be more or less consistent after order 12 hence model order of 12 could be chosen as appropriate order for this system.

D.1.2 Testing with actual data

In previous section, we have used the data acquired by other researchers, however once we had starting recording for our system we performed the same test for model order on our recorded data.

The analysis was done on the data recorded during several sessions for motor imaginary task. The data was acquired in the ideal experimental condition, which includes subject in the setup with a cap on and all the electrodes at the impedance lower than 5kOhms and amplifier parameters were set to the same values as mentioned in the chapter: 4.

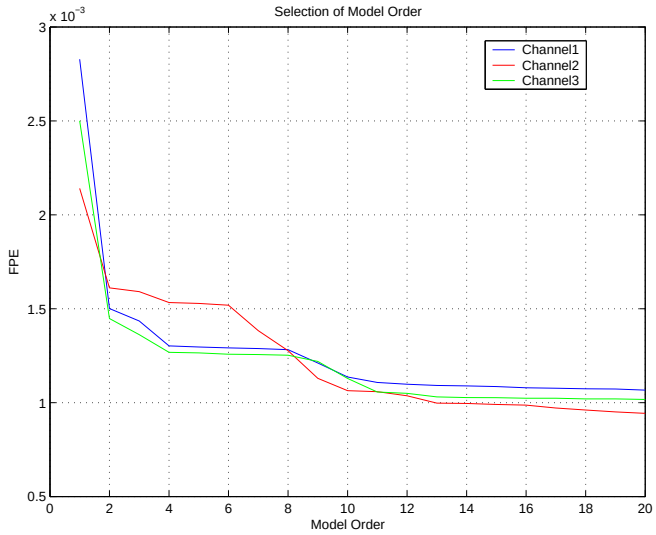

Figure D.1: Functional Prediction Error vs AR Model Order, error seems to be consistent after order 12.

The FPE was calculated for each 1s of data out of 10s for all the 38 EEG electrodes and then averaged over 10s, figure: D.2 shows an example for the same. Then, a mean FPE curve was obtained as averaged over 38 channels and is shown in figure: D.3. A model order is observed for the same. In this particular case, it can be seen that a model order of 9 is appropriate for the system.

This has been repetitively calculated for several 10s trials over different sessions. Using an TSA AR Modeling Order VI from LabVIEW, we obtained the appropriate model order for the calculated FPE. The model order observed from the TSA VI was ranging between 8 to 10 over several trials, hence we decided to have average value 9 as our model order for the system.

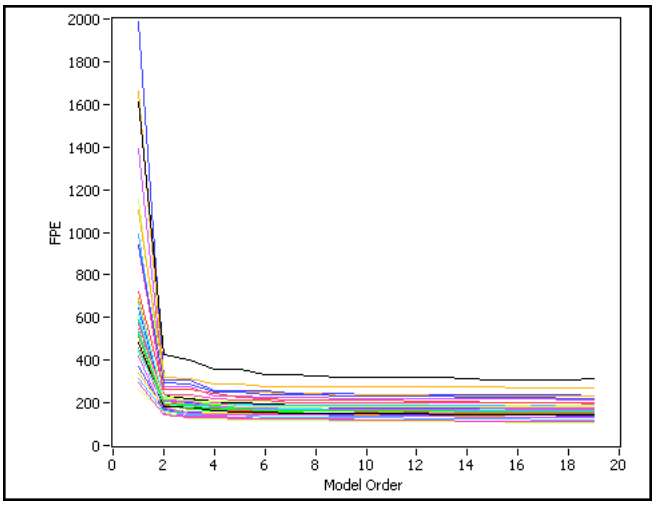

Figure D.2: FPE vs Model Order for 38 channels averaged over ten 1s of data.

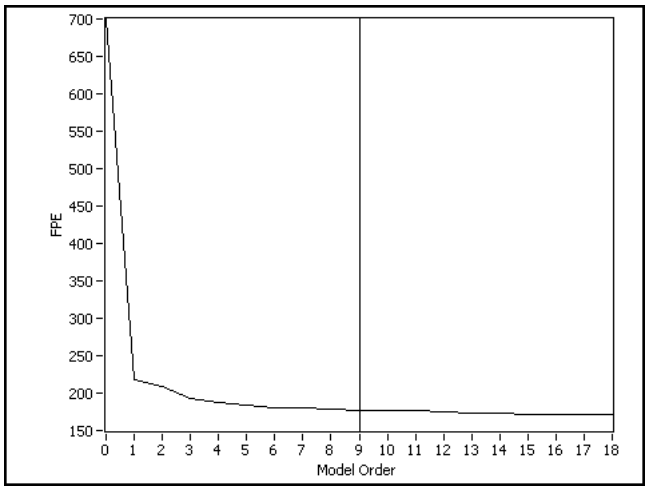

Figure D.3: FPE vs Model Order, mean FPE obtained by averaging over 38 channels.

D.2 Testing of System

Before we could start using BCILab for our sessions we needed to test if we had implemented the system correctly. For this purpose we needed a source that would generate a known and clearly distinctive features. A function generator to help us with that. We set up a normal screening session with a subject, but instead of having the subject trying to imagine a movement for the target given we probed in a 10Hz sinusoid wave of 2mV PtP on either the C3 electrode for a right target or the C4 electrode for a left target. We processed this data in the Offline Tools and could see hugely distinctive features at the expected feature bins with 100% correct classification for left and right. We generated a training file for our 6 selected features with this data and made a feedback session. Again we used the probe to generate signals and had perfect classification for both left and right. We then concluded that our programs and modules were working as desired. The data file for the screening session with the amplifier is called Ampli_1.dat.

Appendix E

I/P and O/P of Modules

This appendix aims at providing the information regarding the various inputs and outputs of the Online and Offline modules used in BCILab. However, for more details about BCILab functionalities please refer: 5 and 6 . All the below mentioned modules can be found in *BCILab.llb* in BCI group directory, SMI, Aalobrg University.

E.1 Online Modules

E.1.1 BCILab Main

It is the main Online module, which performs the data acquisition, signal processing, feature selection and as well as classification. Below discussed are the sub VIs of BCILab main.

E.1.2 Conversion to MicroVolt

Converts the given input into microvolts using the following information:

Inputs

SourceChGain: The gain value of the amplifier.
SourceChOffset : The offset value for the amplifier.
Event Channels: Number of event channels.
Input Signal: Input Signal to the module, *Channels X Samples*.

Output

Output Signal: Signal in microvolt, *Channels X Samples*.

E.1.3 Spatial Filtering

Performs the spatial filtering on all the working channels as well as act as a transmission filter to remove the channels don't need to be processed further.

Inputs

Input Signal: Signal in microvolt, *Channels X Samples*.
Working Channels: Channels need to be sent to signal processing modules .
Not Transmitted Channels: Channels which don't need to be processed.

Output

Output Signal: Spatial Filtered Signal, *Samples X Working Channels*.

E.1.4 Detrending

Performs linear detrending on the given input signal. If desired the method of detrending can be changed.

Inputs

Input Signal: Spatially filtered signal, *Samples X Working Channels*.
Detrending Method: Method used for detrending.

Output

Output Signal: Detrended Signal, *Working Channels X Samples*.

E.1.5 AR Modelling

Generates the feature array for the input signal, by calculating the PSD values of AR coefficients generated from the given method and model order.

Inputs

Input Signal: Detrended signal, *Working Channels X Samples.*
AR Method: Method to be used for AR modelling.
AR Order: Model Order for the system.
AR Frequency Bins: No. of Frequency to be calculated, *(Sampling rate/AR Fequency Bins)* gives the resolution for the spectrum.

Output

Feature Array: PSD values of the working channels, *Working Channels X Frequency Bins.*

E.1.6 Selecting Features


Selects the features provided in the Channel Table (refer figure: 5.2) in the *Parameters* tab and pass those features to the classifier.

Inputs

Feature Array: PSD values of all the working channels, *Working Channels X Frequency Bins.*
Feature Table: Features that need to be selected from the respective channels, column one represents the channel number, *Selected Channels X Selected Frequency Bins.* It reads this table from the Training Data File loaded at the front panel.

Outputs

Selected Features: Selected features arranged in a 1D array with the channels appended one below the other.
No of Features: Gives the number of features selected.

E.1.7 Bayesian Classifier

Performs the classification on the input signal using the selected features from the previous modules.

Inputs

Classification Data: The data from the selected channels and the frequency bins, *Selected Channels X Selected Frequency Bins*.
FLD: Whether FLD needs to be perform or not, *True or False*.
Training: Whether its Training or Classification, in this case (Online) it should be *False*.
W_FLD_in: The *W_FLD* values calculated using FLD operation. It reads this value from the Training Data File loaded at the front panel.
Input Cluster: Contains all the w values calculated during the formation of Training Data File. It reads this values from the Training Data File loaded at the front panel.

Outputs

Error: Gives the error value for the error meter display.
Left ?: Gives *True or False* depending on result code, True is Left.

E.1.8 Training File Reader


It reads the Training File, which is formed during the Offline Training stage.

Inputs

Input File: Path to the Training File.

Outputs

FLD: Whether FLD was performed while computing Training File.
Output Cluster: Contains all the w values calculated during the formation of Training Data File.
FLD Values: The *W_FLD* values calculated using FLD operation.
Channel Table: Features used to compute the Training File *Selected Channels X Selected Frequency Bins*.
Best Left Error: The highest g value for the left target.

Best Right Error: The highest g value for the right target.

E.2 Offline Modules

E.2.1 Feature Search

 It is an main Offline module, which performs feature selection and generates training file to be used in online module. Below discussed modules are sub VIs of Feature Search.

E.2.2 Feature Reader

Reads the feature file and put the information in a cluster.

Input

Feature File Path: Path to the feature file.

Output

Feature Cluster Array: Cluster containing all the information in the feature file.

E.2.3 Cut Feature Space

 Cuts the feature space according the parameters in Feature Table.

Input

Feature Cluster Array In: Feature array containing all information regarding feature space, target code, selected channels.
Limited Feature Space ?: *True* if the offline analysis needs to be done on selected channels and feature bins else *False*.
Selector: Feature table with the channel number and frequency bins information on which analysis needs to be done.

Output

Feature Cluster Array Out: Contains all the information as that of Input array but only for the selected channels and feature bins.

E.2.4 Train & Test

Gets *(k-1)* subgroups for both state 1 and 2 and train the classifier with that data. The it classifies a single subgroup of either state.

Input

Training Array Class 1: Data to train the classifier for state 1.
Training Array Class 2: Data to train the classifier for state 2.
Classification Array Class 1: State 1 data to be classified.
Classification Array Class 2: State 2 data to be classified.
FLD: If FLD is in use or not.

Output

Success Rating Class 1: The percentage of the state 1 data that got classified correct.
Success Rating Class 2: The percentage of the state 2 data that got classified correct.

Appendix F

Setup of the 64 Channel Electro Cap

F.1 Introduction

This document describes how to connect and setup the 64 channel EEG Electro Cap. The cap in reality does not contain 64 EEG electrodes but has 59. These 59 electrodes are then connected to wires that are bundled into two cables. One cable (referred to as Cap Connector 1) contains 31 wires while the other (referred to as Cap Connector 2) contains 28 wires. Both end in a 37 pin connector plug. The electrode/pin connection can be seen in the following picture F.1 and F.2 for both connectors.

F.2 Rewirering Connectors

Two NuAmps amplifiers made by Neuroscan are used to record from the cap. Each amplifier has 40 available channels for recording and are also equipped with a 37 pin connector plug. Sadly, even though the connectors from the cap can go directly into the plug on the amplifiers this will not work. The channel layout of the connector plug and the amplifier plug does not match. Therefor it is required to rewire the channels from the cap plug directly into the 40 inputs on top of the amplifiers. We have created two connector cables for this purpose. Due to some restrictions in the Acquire software about how to label channels in multiple amplifiers it is required to wire 7 channels from Cap Connector to Amplifier 1 and the rest to Amplifier 2. The setup can be seen in the following, see figure: F.3.

This also shows that the two reference channels (A1 and A2) has to be connected to both amplifiers. Ground also needs to be connected. The electrode labeled "GND" can be used for this purpose. Normally this electrode would be connected to Amplifier 1 in channel port 37 but to use it as ground is has to be put in one of the "GND" channel ports on either amplifier.

The channels has to be connected to the amplifiers in the following order, see figure: F.4.

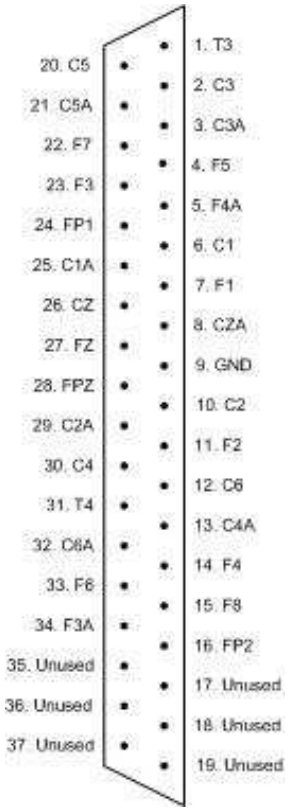

Figure F.1: The Cap Connector 1.

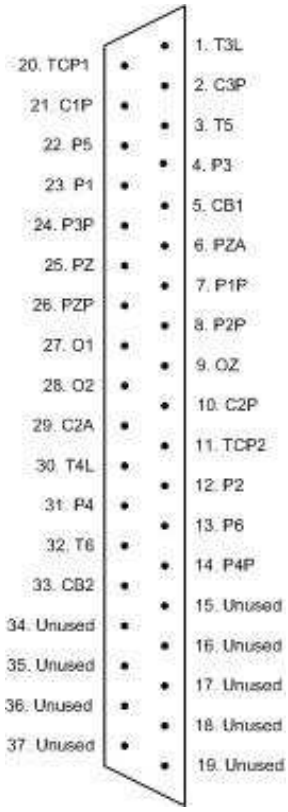

Figure F.2: The Cap Connector 2.

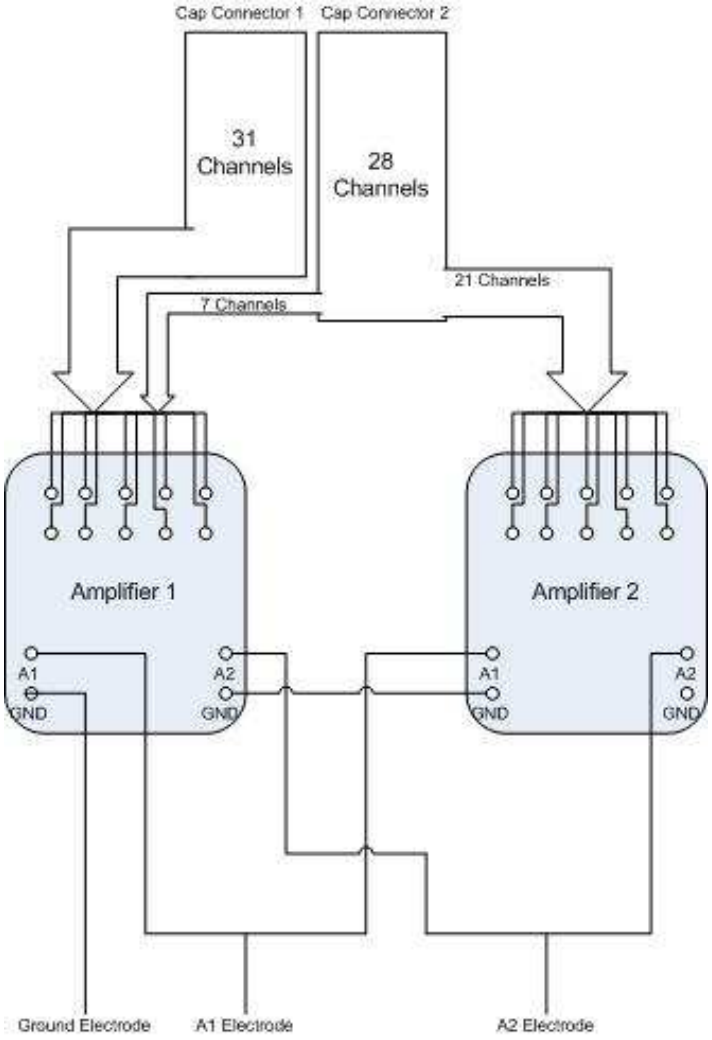

Figure F.3: The Connection Setup.

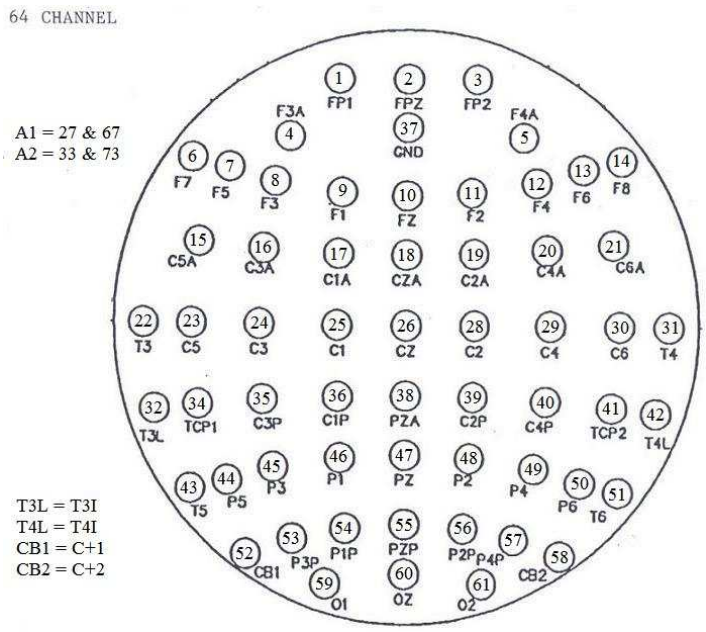

Figure F.4: 59 channel setup.

Due to unavailability of the correct letters some of the connector plugs had a small renaming. Their alternative name is found in the lower left part of the picture. To get the physical channel numbers for the second amplifier just subtract 40 from the numbers on the picture.

Both amplifiers should be connected to a USB hub and the hub then connected to a computer. It can take a little while before the computer recognizes both amplifiers and install their drivers.

F.3 Acquire Setup File

We have produced at channel setup file that is to be used in the Acquire software program to give the correct labeling of the channels. The name of the setup file is NuAmps_59_64chan_Electrocap.ast. There seem to be a mix up of channel A1 and A2 on the second amplifier. Even tough they are connected to the physical channels 67(27) and 73(33) on the second amplifier and the Channel Assignment Table verifies that, their values are actually recorded at channel 62(22) and 63(23). This means that it is not possible to verify low impedance on the reference channels for the second amplifier. It is not advised to put the connector plugs for A1 and A2 anywhere else than 67 and 73 as those are still being used as reference.

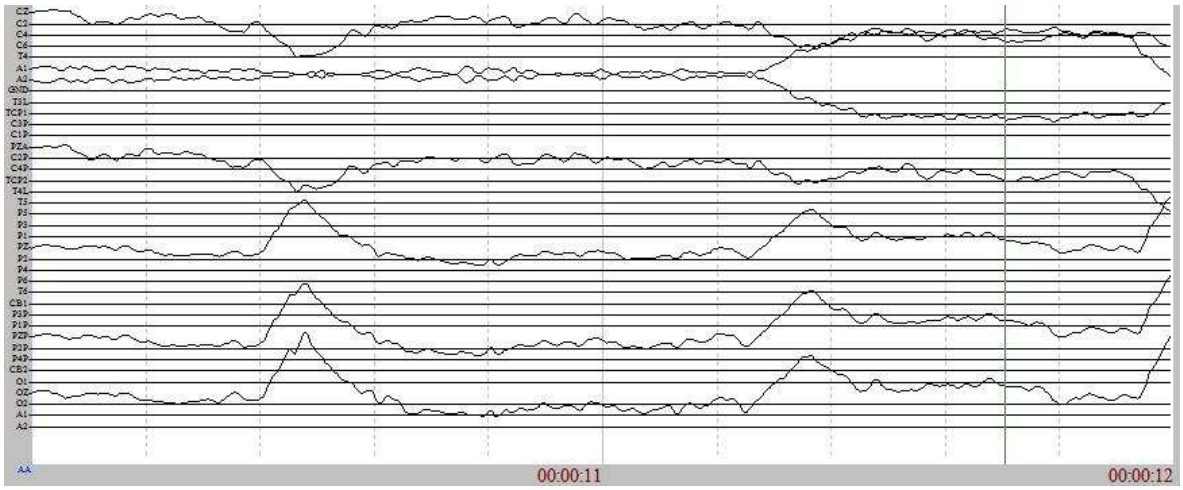

Figure F.5: Trace of a blink pulse.

F.4 Identifying Problems

We tested our setup and encountered some serious problems. All our testing was done on a laptop running on battery and all electrical devices in the room turned off. The sampling frequency is 250 Hz. Firstly, it we observed in figure: F.5 some phase problems. As seen in the following picture of a blink movement, it seems like there is a change of phase in the two amplifiers.

When signals from one amplifier increases they decrease on the other. Secondly we observed in figure: F.6, transients in the channels on the second amplifier. These transient came randomly with a very characteristic shape. We tried to change the order of the amplifiers but the "second" amplifier would still give the same problems. We contacted Neuroscan about these problems but have not gotten any reply till the time of writing this.

F.5 The 40 Channel Setup

Due to the problems described previously and to lessen complexity of our system we decided to reduce the number of channels to record on to 38 and with the A1 and A2 channels having 40 channels all in all. Thereby we would only need one amplifier in the setup.

The channel layout for the 40 channels are seen in figure: F.7.

It should be noted that it is still required to connect channels from both cap connectors to the amplifier. The ACQUIRE setup file for this setup is NuAmps_59_40chan_Electrocap.ast.

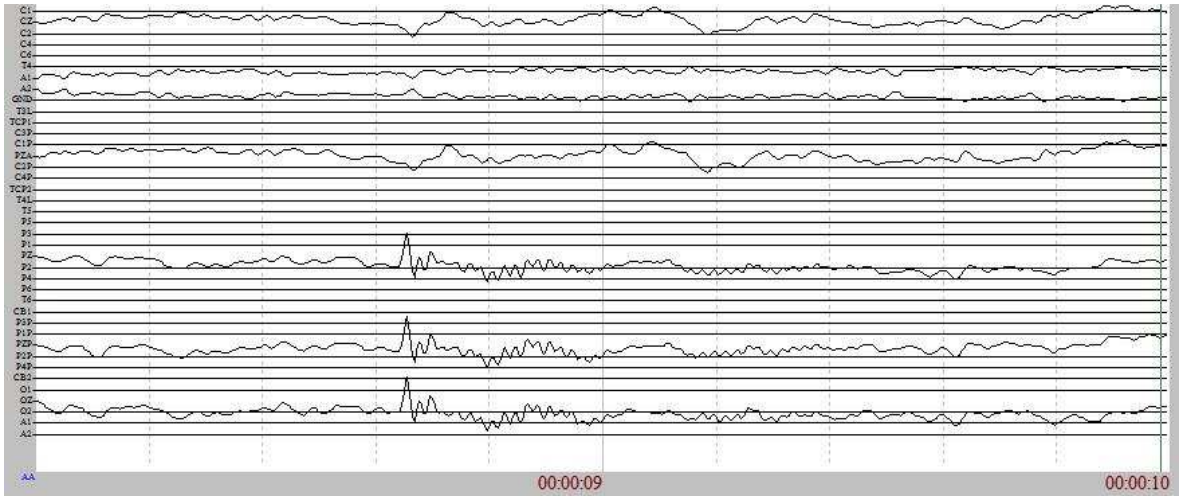

Figure F.6: Signal transient.

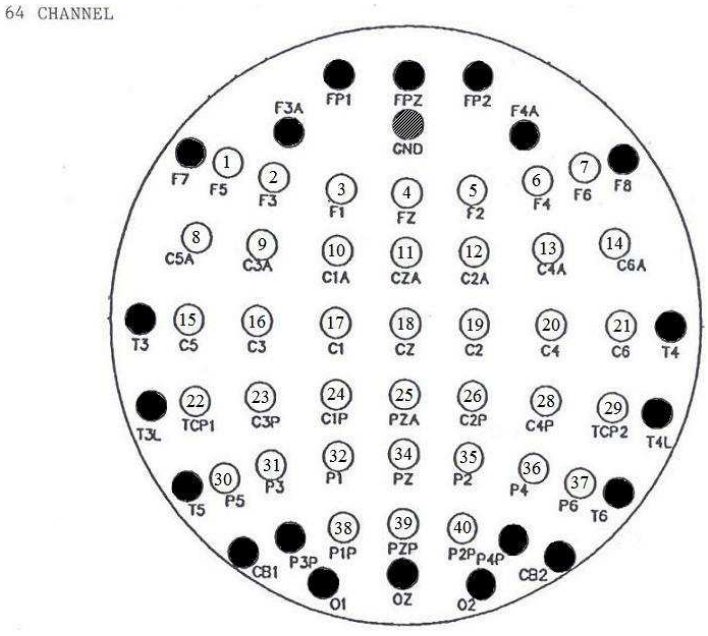

Figure F.7: 40 channel setup.

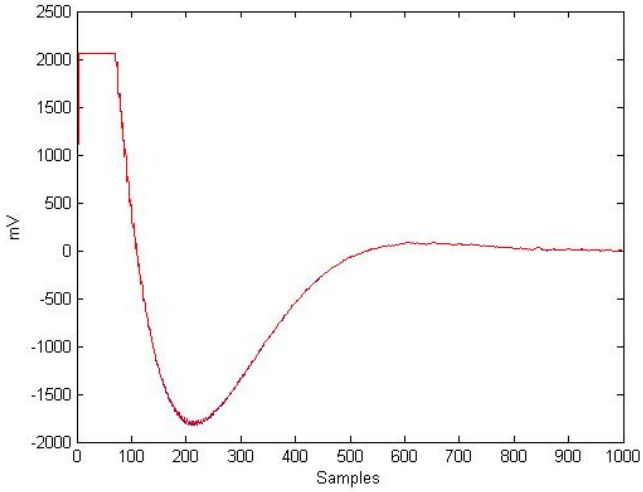

Figure F.8: Tune in stage.

F.6 Tune In Stage

To verify that our system actually is recording the same as ACQUIRE we performed a comparison test. Figure: F.8 shows a such for a single channel. The figure is made in MATLAB. It can be seen that the two lines (red and blue) nicely follows each other. The recording is of the tune in stage of the channel. That is, the behavior of the channel in the seconds after the amplifier has been turned on. A huge and slow change in values can be observed until the system goes to steady state around zero volt after a few seconds. The sampling frequency is 250 Hz. The tune in stage is not considered a problem for our screening and feedback sessions as it normally takes more than a few seconds between starting the amplifiers and starting the recordings.

F.7 Adding EMG Electrodes

We added 6 channels more for EMG recordings. The setup file for this is NuAmps_59_40chan_Electrocap_EMG.ast.2 channels record from the left arm, 2 from the right and 2 around the left eye. As we now again have more channels than there can be on one amplifier the second amplifier is taken into use again, but this time only to record EMG. Also, as impedance level between the ground electrode for the EEG channels and the EMG channels is far too high we made a separate ground on the ankle of the left foot for the EMG channels on the second amplifier. This means, that even though the 6 EMG channels can be seen on the Impedance level screen in ACQUIRE; their level will not decrease even when connected properly. Instead the Data Display should be started for inspection of the channels. The final setup with the two amplifiers can be seen in figure: F.9.

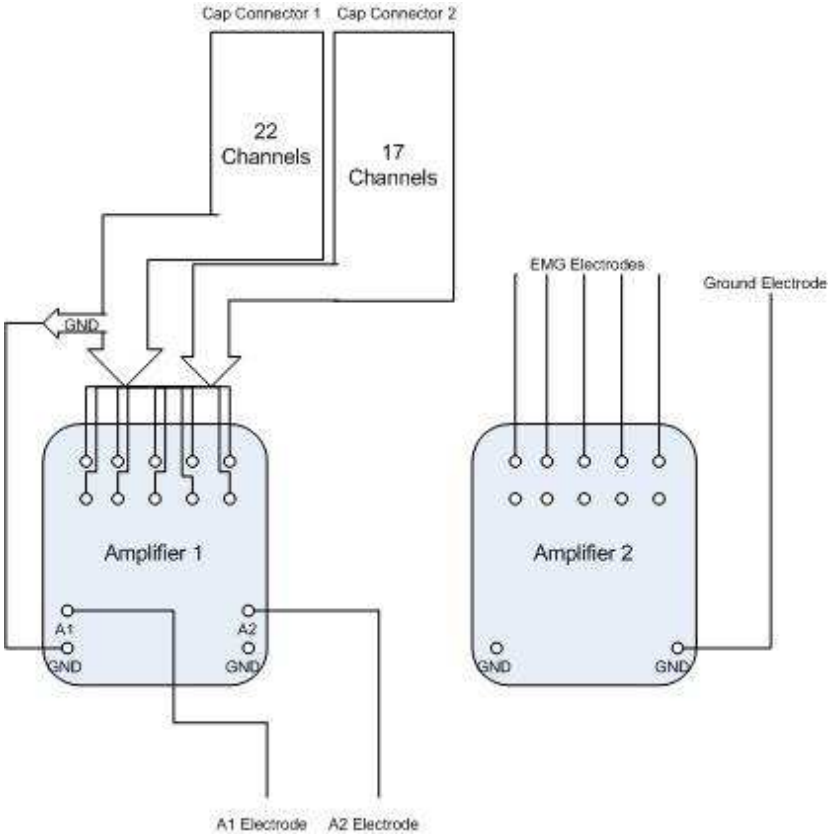

Figure F.9: Setup with EMG channels.

Appendix G

Data Storage File

This is an example of a the header of a storage file for state definitions and parameters following the BCI2000 standart. Parameter format is:

'section' 'data type' 'name' 'value' 'default value' 'low range' 'high range' // 'comment' CRLF

All parameters are being described by their comment. For more information about the format consult the BCI2000 documentation [16].

HeaderLen= 3486 SourceCh= 46 StatevectorLen= 4
[$StateVectorDefinition$]
SourceTime 16 0 0 0
Running 1 0 2 0
Recording 1 0 2 1
TargetCode 2 0 2 2
ResultCode 2 0 2 4
ErrorRatingSign 1 0 2 6
ErrorRating 8 0 2 7
[$ParameterDefinition$]
System int SessionType= 2 1 1 2 // The type of recording session (1: Screening, 2: Testing)
UsrTask int NumberOfTrials= 10 1 1 100 // The number of trials Source int SoftwareCh= 46 38 1 64 //the number of received and stored channels
Source int SampleBlockSize= 250 250 125 1000 //the number of samples transmitted at a time (sample frequency)
Source int TransmitCh= 38 38 1 64 // the number of transmitted channels
Source intlist TransmitChList= 38 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 38 1 64 //list of transmitted channels
Filter int SpatialFilteredCh= 38 38 1 64 // the number of channels spatially filtered
Filter intlist SpatialFilteredList= 38 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 38 1 64 //list of spatially filtered channels
Source int SamplingRate= 250 250 125 1000 //the sample rate (sample frequency)
Storage string SubjectName= Asbjoern Name a z //subject alias Storage string SubjectSession= 1 001 0 999 //session number (max. 3 characters)

Storage string FileInitials= C:\Documents%20and%20Settings\Alvaro%20F.%20Cabrera\Desktop\06GR985
05-07 a z 0 //Initials of file name
Storage string FeatureSpaceFile= C:\Documents%20and%20Settings\Alvaro%20F.%20Cabrera\Desktop\06
05-07\AB_1_F_featsp.txt a z 0 //Name and path of the features space storage file
Storage string TrainingDataFile= C:\Documents%20and%20Settings\Alvaro%20F.%20Cabrera\Desktop\06
05-07\AB_1_Sr_trainingdata.txt a z 0 //Name and path of the training data file
Filtering floatlist SourceChOffset= 46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -500 500 //offset for channels in A/D units
Filtering floatlist SourceChGain= 46 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 -500 500 //gain for each channel (A/D units -¿ muV)
UsrTask int TargetSizeLeft= 3600 1 1 10000 // Size of the left target indicator (in pixels)
UsrTask int TargetSizeRight= 3600 1 1 10000 // Size of the right target indicator (in pixels)
UsrTask int ErrorDisplaySize= 17856 1 1 10000 // Size of the Error Meter display (in pixels)
UsrTask int SuccessDisplaySize= 4408 1 1 10000 // Size of the Success Rating display (in pixels)
UsrTask int RewardDuration= 5 3 1 100 // The time for reward (in seconds)
UsrTask int PreTrailPause= 5 5 1 100 // The time for between the target light up and recording starts (in seconds)
UsrTask int RestingPeriod= 7 7 1 100 // The time for rest between trials (in seconds)
UsrTask int TrialTimeLength= 10 10 1 100 // The recording time (in seconds)
Filtering string DetrendMethod= Linear Linear a z // The detrend method
Filtering int AROrder= 16 16 1 20 // The order of the AR filter
Filtering int ARFrequencyBins= 125 125 1 1000 // The number of frequency bins for the AR filter
Filtering string ARMethod= Burg-Lattice Linear a z // The detrend method System int StateVectorLength= 3 3 1 30 //length of the state vector in bytes ...followed by the binary data...