

Towards a fast spelling device for a Brain
Computer Interface:
the quest for a third signal source in the EEG

Auke L. Klazema

June 23, 2010

Internal supervisor: prof. dr. L. Schomaker

External supervisor: prof. dr. R. de Jong

Artificial Intelligence
University of Groningen

Abstract

A brain computer interface (BCI) would provide “locked-in” patients that have lost all voluntary muscle control the ability to communicate again with their surrounding. By controlling the computer only with brain activity the study set out to create a faster spelling device than the Hex-o-Spell. Traditionally, imagined movement is used to create control in an EEG-based BCI system. Usually, these are thoughts about left and right hand movements. The Hex-o-Spell application from the Berlin BCI group was modified in the current study by adding a third thought (imagination) to the CSP-based classification. Unfortunately the Berlin BCI record of 7 character per second wasn’t broken because of project time constraints but a promising third brain signal was found to create a faster spelling device for BCI: the beta rebound is reliably detected in one subject and further study needs to confirm the usability of this signal.

Acknowledgements

Science is never done by one individual, it's a very social line of work. People use each other findings to take the next step in creating a better picture of nature.

I also didn't do it all on my own but had help and support from many different persons. I want to say something about these persons in this part of the thesis because without them I wouldn't be able to finish it in the way I did.

First of all I want to thank my external supervisor prof. dr. R. de Jong in guiding me on regular moments with the project. We had many hours of discussions on BCI research that resulted in an interesting project and cooperation.

The next person on my list is Mark Span who was the technical advisor on the project and helped me out with the EEG lab and the BCI2000 program. We also had many moments of fun during our breaks.

I also wish to thank my internal supervisor prof. dr. L.R.B. Schomaker for guiding me in the beginning of the project and helping me in the end with the writing of this thesis.

Stella Banis for teaching me the in and outs of applying EEG Electrodes. But also for our conversations about both her and my wives pregnancy.

And most importantly my wife for the mental support and patience during my many years of studying on the university that took many hours of our time that we could have spend together.

Thanks

Contents

Abstract	ii
Acknowledgements	iii
Contents	iv
1 Introduction	1
2 Overview	3
2.1 Other BCI systems	4
2.2 Beta-rebound	5
2.3 Research question	5
3 Technical and theoretical background	7
3.1 Brain activity	7
3.1.1 Neurons	8
3.1.2 Somatomotor cortex	9
3.1.3 Brain waves	10
3.2 EEG	11
3.3 Common Spatial Patterns	12
3.4 K-fold cross validation	13
3.5 Fisher Linear Discriminant	14
3.6 Tools used	15
3.6.1 Electrodes and Cap	15
3.6.2 Amplifier	16
3.6.3 Computer & Software	16
3.6.4 Matlab	17
3.6.5 BCI2000	17
3.6.6 Experimentation Room	19
3.7 Subjects	20
4 Hex-O-Spell	21

4.1	Interface	21
4.2	Language model	22
4.3	Classification	23
4.4	Steering	23
5	TriSpeller	24
5.1	Interface	24
5.2	Language model	25
5.3	Classification	25
5.4	Steering	26
5.5	Gray area	26
5.6	Bias adaptation	27
5.7	Testing	27
5.7.1	Results	27
5.7.2	Discussion	29
6	Pilot experiment	30
6.1	Basic paradigm	30
6.1.1	Task	30
6.1.2	Classification run	31
6.1.3	User training run	31
6.1.4	Experiment run	32
	Stimuli	32
6.1.5	Results	32
6.1.6	Discussion	32
6.2	Selecting letters	33
6.2.1	Experimental task	34
6.2.2	Results	34
7	Discussion	35
A	Matlab Code	37
A.1	bci Initialize.m	37
A.2	bci Process.m	38
A.3	bci StopRun.m	41
	Bibliography	51

Chapter 1

Introduction

“Because we do not understand the brain very well we are constantly tempted to use the latest technology as a model for trying to understand it. In my childhood we were always assured that the brain was a telephone switchboard. (‘What else could it be?’) I was amused to see that Sherrington, the great British neuroscientist, thought that the brain worked like a telegraph system. Freud often compared the brain to hydraulic and electro-magnetic systems. Leibniz compared it to a mill, and I am told some of the ancient Greeks thought the brain functions like a catapult. At present, obviously, the metaphor is the digital computer.”

John Searle

People have a fascination for things that change. This is also the case for scientists, for studying something that doesn’t change is very hard. There are no variables to manipulate to deduce it’s workings. The brain in general and the human brain in specific have always been of great interest to science. It is so complex and capable of so much yet only a fraction is known of its working.

The brain controls the environment through the use of its body that contains it. It is very in need of the input and output it gets from its body. If it didn’t have a body it could only process its own input and output without changing it’s environment. The reason that I quoted a famous philosopher of artificial intelligence, i.e., Searle, is because I too want to use the analogy of the computer to talk in a more understandable way about the brain and its body.

The brain would be equal to the Central Processor Unit (CPU) and the body would be equal to the Input and Output (I/O) modules (e.g. the sound-card or keyboard). So when we look at the brain as useless when it has no body so too is the CPU useless without the I/O modules. Luckily most brains do have a functional body that allows the brain to interact with its environment. But there are people who have lost most of their I/O and with it the ability to communicate. So called locked in patients.

The field of Brain Computer Interface (BCI) is a research field that tries to add a new Output possibility for the brain. It uses the information that can be collected from the brain to do something useful in the world. Along the way science learns new facts about the brain because it can change input variables and detect the changes in the brain and in a sense reverse-engineer the brain.

This master thesis describes the attempt to create an useful application by means of a brain-computer interface. A spelling device was created that allows the locked in to communicate with the world around them. BCI spelling devices have been created before, but we wanted to create one that would have a higher spellings rate. Unfortunately six months weren't long enough to create a working device with a high spelling rate but new things have been discovered and this thesis shows a promising new way to create a faster spelling device.

This report is intended to be a good document of what has been done in these six months but more specifically how it was done so others could continue the work. Hopefully it has been written in a way that is understandable by more students than just the students of Artificial Intelligence. This is because the research field is of a multidisciplinary nature. I tried to detail all the essential parts of our BCI system such that others could replicate it. I hope I have succeeded.

Chapter 2

Overview

A brain computer interface collects brain activity and finds patterns that can be classified by the computer which then does something useful with this information. Brain activity recording techniques can be divided into two groups; invasive and non-invasive. Invasive techniques collect the activity inside the skull and the non-invasive collect it on the outside of the body.

Finding patterns in the brain signals has been done in different ways. The first BCI systems used specific templates (preprogrammed patterns) where the user needed to mold his or her brain waves to let the computer do something useful [Birbaumer et al., 1999]. The newer systems let the user think specific thoughts and the system will determine the templates all on their own [Wolpaw and McFarland, 2004].

To let the user know if the computer was correct in detecting their thoughts some kind of feedback is needed. This is traditionally done with visual feedback but more and more different kinds of feedback systems are used.

BCI systems are closed loop systems (Figure 2.1) in which it is very difficult to change variables and hold all the others stable. The human brain is continually responding to the feedback it receives. It might even be adjusting itself during the time the BCI is being modified or even during the testing of the system. This can make it very difficult to be sure that a modification was the cause of a better classification score.

Developing a system can be a slow process because it takes time to prepare for a testing. Software needs to be adjusted, data needs to be analyzed and subjects need to be found that have time to be tested when the system is ready. Because of the closed loop, recorded data can't be used to test the entire system because the feedback is depended of the EEG but the EEG is also depended of the feedback. The only thing that can be modified and tested offline is the classification algorithm. But to be sure the

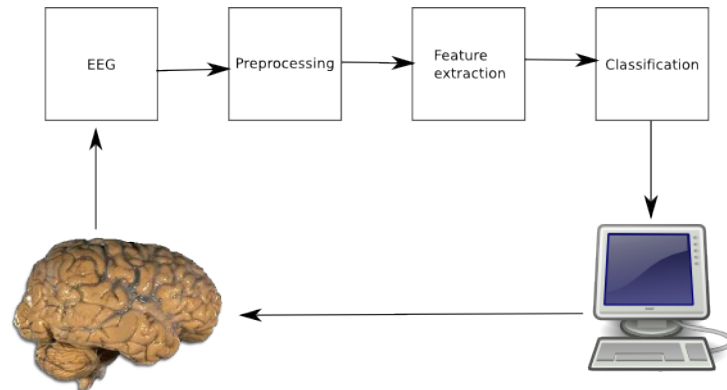


Figure 2.1: The BCI closed loop with all the parts that are involved. The brain activity is being collected by EEG. The Collected data is then pre-processed that usually includes filtering. Within this preprocessed data, features are extracted. These features are then used to classified the brain activity. The classification is used to present the brain with feedback. In response to the feedback the brain generates new brain activity.

modification actually worked better, testing with a real person still needs to be done.

For a good overview paper see [Wolpaw et al., 2002]. Maybe it is starting to show its age but it still covers all the basics of a BCI system.

2.1 Other BCI systems

The system discussed in this thesis used CSP with EEG data applied to motor activity but there are other ways of building BCI systems. The invasive techniques were already mentioned, where electrodes are placed directly into the brain [Leuthardt et al., 2004]. Because BCI systems are build with modules as seen in figure 2.1 the different kind of systems will be discussed according to these modules.

The collection part that is called EEG in figure 2.1 can be replaced with for example: MEG [Mellinger et al., 2007], fMRI [Weiskopf et al., 2004] or NIR [Coyle et al., 2004].

A selection of signal types that can be detected are Steady State Visual Evoked Potential (SSVEP) [Middendorf et al., 2000], Slow Cortical Potentials (SCP) [Birbaumer et al., 1999], P300 evoked potentials [Sellers and Donchin, 2006, Piccione et al., 2006], sensorimotor cortex activity.

Almost all of the non-invasive BCI systems have a one dimensional con-

trol with the exception of [Wolpaw and McFarland, 2004]. They were able to train subjects to control a cursor in two dimensions. This took many hours of training and selection of subjects within a large pool was necessary because not everyone was able to get 2D control.

Traditionally visual feedback is given, but this might not always be the best feedback type for locked in patients who have lost voluntary control of their muscles including the eye muscles. Other feedback types have been investigated. Auditory feedback [Nijboer et al., 2008] would not depend on the muscle activity and probably be better suited for locked in patients.

Tactile feedback has also been researched [Cincotti, 2007]. Both the tactile and auditory feedback can be just as good as visual feedback but more training may be needed.

The patterns in brain activity can be found with the use of machine learning techniques. CSP with a linear classifier is just one of the techniques that had good results in the BCI research field but other work as well. For example Neural Network, Bayes and Nearest Neighbor [Lotte et al., 2007].

2.2 Beta-rebound

Beta rebound is a brain signal that is detected by EEG. It has been associated with starting and stopping movement.

The beta rebound has been a signal that is well documented and the university has experimented with it before. The beta rebound has been found by using real and imagined movement [Pfurtscheller et al., 2005] [Pfurtscheller and Solis-Escalante, 2009]. It needed to be tested if it also existed when only imaged movement was used in the TriSpeller application. The beta rebound is a burst in the beta frequency that is strong and short. What the function of the beta rebound is, is not known but it has something to do with movement.

It was hypothesized that when changing from one movement to another the beta rebound is more profound than when the movement is just stopped.

2.3 Research question

This study wanted to create a BCI spelling device that works better than the BCI Hex-o-Spell by using new techniques to increase the amount of detectable brain states. This will be measured in character output per minute.

The following techniques will be used to meet this target: motor imagery and beta rebound. Also the information available within the system that can be useful to determine the intentions of the user will be used.

The university of Groningen has been working with motor imagery before with good results. This will be used as a basis for the current study.

The research question then becomes: Does the character output increase when beta rebound gets used in combination with motor imagery within a Brain-computer interface? Can a beta rebound be detected real-time and used as a signal for control of a BCI?

Chapter 3

Technical and theoretical background

This chapter explains the basic technical and theoretical issues behind the TriSpelle BCI system. Some of the choices made will be explained here but others can be found in other chapters.

3.1 Brain activity

The brain consists of about 100 billion neurons that all work together to create a highly complex system that allows us to do what we do. The neurons communicate with each other through chemicals and through electrical signals. When a region of the brain is more active, more of this communication

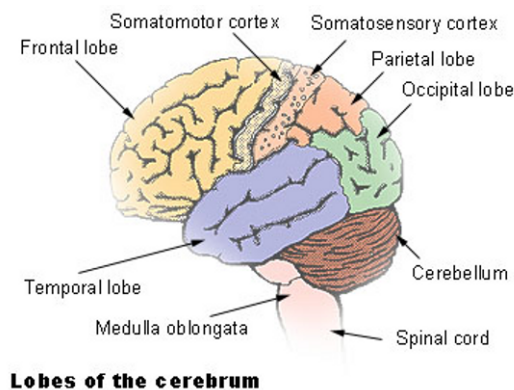


Figure 3.1: Brain lobes overview with the somatomotor cortex located in the middle of the brain.

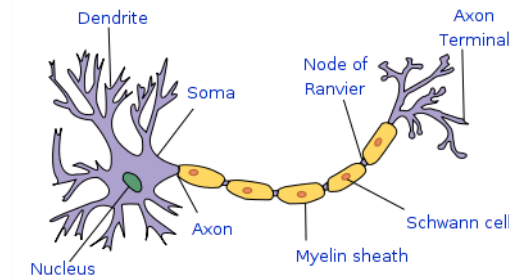


Figure 3.2: The neuron is the basic building block with a cell body that gets extended by an optional axon and dendrites.

is going on [Kalat, 1998].

The region of the brain that has been used in this thesis is the so called somatomotor cortex (Figure 3.1). This section explains how neurons produce electrical signals following with a description of the somatomotor cortex.

The activity comes in different rhythms. Different functions of the brain can have different rhythms and these are divided into groups and are called brain waves. The TriSpeller BCI system depends highly on these brain waves so they are also described in this section.

3.1.1 Neurons

The neuron is the basic building block for the brain. They create a large interconnected network that communicate with each other through electrical signals and chemicals. The neuron consists of a cell body, dendrites and optionally an axon.

The cell body is responsible for the energy production and the maintenance.

Dendrites are the extensions of the cell that collect signals from other neurons. A single neuron can have a few dendrites but some type of neurons can have more than 100.000 dendrites. The axons and dendrites communicate through neurotransmitters.

These neurotransmitters connect to dendrites and will open ion conducting channels which will change the ratio of the negative and positive ions inside and outside of the neuron. This can either increase or decrease the difference potential of the neuron.

The axon sends electric pulses based on input from dendrites, chemicals and the state of the neuron. The neurotransmitters can raise the normal resting potential of about -70 mV slightly. When this potential is raised

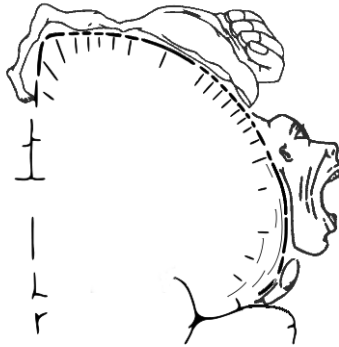


Figure 3.3: The homunculus shows the specialization of the cortex for different muscle groups. It also shows clearly that the control of the movement of hands and arms take a large portion of the brain. This is why hand movements is mostly used in BCI research.

beyond the threshold of about -50 mV an action potential will be fired. This reaction is called the all or nothing principle.

When the threshold is reached the neuron will open ion conducting channels which will increase the differences in positive and negative ions very rapidly. The action potential will finally end up at the end of the axon. Here neurotransmitters will be released.

This action potential can be measured with micro probes but they can also be picked up on the outside of the brain when enough neurons fire at somewhat the same time yielding a time-varying electrical field that can be measured.

3.1.2 Somatomotor cortex

This part of the brain gets active when muscles get activated. The cortex is divided into sections that each control a different part of the body. A big division is made into left side control and right side control of the body. The left hemisphere is responsible for the right side of the body and vice versa.

When only the thought of movement is created without the actual movement, activity is found in the same place as when the movement is actual done. The difference is that the signal is a little weaker but it is detectable [McFarland et al., 2000].

The hands take a relatively large part of the brain (Figure 3.3). This is because good control in the hands is very important and it takes more

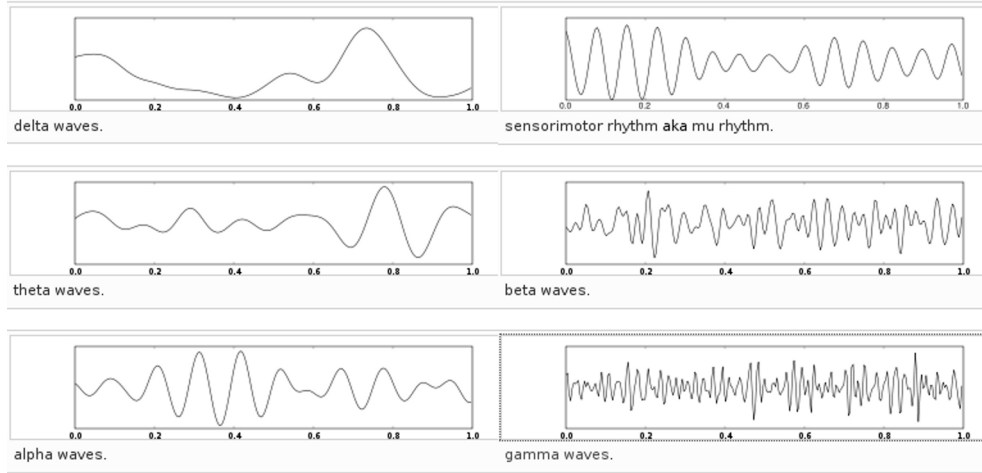


Figure 3.4: Different type of EEG waves each belonging to a frequency group

brain power to make this possible. When a big motion is made, more of the cortex is activated and thus better detectable (depending on the position in the brain) by EEG.

Depending on the position of the neurons in the brain the firing can be measured on the outside of the scalp. Hand movement as well as foot movement are reasonably well detectable with EEG. With other recording techniques other movements can be detected even better than EEG but these have disadvantages as well.

3.1.3 Brain waves

The brain activity comes in different frequencies (Figure 3.4). This happens when multiple neurons fire at the same rate. These firings combined are then visible in the spectrum of the EEG. For the experiments the mu frequency region is of importance as well as the beta region.

The brain waves have been categorized into frequency ranges. This make for easy communication but they also correspond to certain forms of activity in the brain.

Delta waves run up to 4Hz and are associated to slow sleep.

Theta waves run from 4 to 7 Hz. They are related to a drowsiness state of the brain in where the brain becomes more idle and almost asleep.

Alpha waves run from 8 to 12 Hz. They are found when the brain is in a relaxed state.

Beta waves run from 12 to 30 Hz and are seen in the EEG when the brain is active and alert.

Gamma waves run from 30 to 100+ Hz. These waves have been associated with cognitive and motor activity but it is a difficult range to detect correctly because of muscle activity.

The two ranges that are of interest to the current study are the Mu and the Beta ranges.

The mu region runs from 8 to 12 Hertz (Alpha). This activity is associated with the actual movement taking place in the somatomotor cortex. So when a person thinks of either the left or right movement it is expected that a difference would be best found in the mu region.

The beta region used in the study runs from 15 to 30 Hertz. This frequency range shows activity after a movement is stopped. Or when one is stopped and another gets started. A complete theory behind the beta activity related to movement has not yet reached consensus in the research community.

3.2 EEG

Electroencephalogram (EEG) is a brain activity measurement technique that takes place outside of the body. The measurements are done on the scalp through the use of electrodes. These electrodes pick up difference in potential. This is measured against a ground electrode that is usually placed on the chest.

The scalp functions as a sort of conductance plate that smears the electricity. This makes it hard to be sure that the electrical signal that gets measured at a certain location is also coming from that location.

There are techniques that reduce this smearing and overflow of activity by subtracting neighboring electrodes [Wolpaw and McFarland, 2004]. New techniques also use head and scalp models to determine more accurately where the signals are coming from [Pascual-Marqui et al., 1994].

The big advantage of EEG as opposed to other recording techniques is the high temporal resolution. This means that real-time recording can be done. There are no calculations needed the electrical signals can be used instantly. The big drawback is the low spatial resolution as mentioned earlier.

Most of the research done with EEG where new signals were found used multiple recordings that get averaged. This results in a signal that is clear and clean of most noise. BCI research doesn't have the possibility of averaging these signals because it would take too much time which would

render the BCI system useless.

New techniques need to be created to detect states in real-time and on a single signal. The system needs to deal with the extra noise the EEG amplifier picks up.

3.3 Common Spatial Patterns

This section describes the main classification algorithm.

Because filtered signals were used, there is no difference in group means between the two imagined movements, so a different way had to be used to do statistical analysis. CSP doesn't use means but spatial covariance to determine differences between the groups.

Raw EEG data has a low spatial resolution caused by the volume conduction. By using a spatial filtering this resolution is raised. With this weaker signals can be better detected because stronger signals from elsewhere are filtered out. The motor signals are very weak and that's why CSP is a good technique for detecting and differentiating left- and right side motor signals.

The algorithm starts with a matrix $C \times S$ where C is the number of EEG channels that are used and S the number of samples per channel. Next it calculates the a normalized spatial covariance for each hand with the following formula:

$$C = \frac{EE'}{\text{trace}(EE')} \quad (3.1)$$

The spatial covariance will be averaged over all trials. After this is done the composite spatial covariance is calculated with the following formula where l and r stand for left and right:

$$C_c = \overline{C}_l + \overline{C}_r \quad (3.2)$$

Now when the simultaneous diagonalization gets calculated it results in a matrix W filled with generalized eigenvalues that solve the generalized problem:

$$\overline{C}_l w = \lambda \overline{C}_r w \quad (3.3)$$

Where λ is:

$$\lambda = \frac{\lambda_l}{\lambda_r} \quad (3.4)$$

$$\lambda_l + \lambda_r = 1 \quad (3.5)$$

Then when the highest λ gets selected with the corresponding w the variance will be higher for the left condition and lower for the right condition. When the lowest λ gets picked the situation will be reversed. This result can be used in a linear classifier to create a good classification between left and right conditions.

Finally the W' can be used to multiply with the EEG signals resulting in a signal that can be used to train a linear classifier to later create predictions about the thoughts of a person.

3.4 K-fold cross validation

To test the classification system of the BCI K-Fold cross validation was used [Cohen, 1995]. This is a method to train and test classification algorithms by using a part of the collected data for training and the other part for testing.

It starts out with shuffling the collected data. With this random collection K equal sized parts are created. One of the parts is used to test the system and the remaining parts are used to train the CSP and the Linear Classifier. This is done K times each time a different part is used to test the system.

With every new training the CSP and the Linear Classifier needs to be retrained. If this is not done the system learns incorrect information and some data is then over trained but also the part that needs to be tested is already included in the training data in the previous training.

When all the K training and test sequences have been performed the average result will be calculated and returned as the final answer of how good the CSP and Linear Classifier performs on the collected data. It also shows how much the collected data looks alike and gives some confidence that about the usability of the collected data. If the result was very low then the collected data is not very clean and clear in the ability to distinct the two brain signals. So in a sense it also gives an indication of how good the data is for calculating a CSP filter that is of high quality.

With this information the experimenter can decide to retrain the same movements, change the movements, use the data or decide that the subject is too difficult to train in the selected movements.

3.5 Fisher Linear Discriminant

The fisher linear discriminant analysis can be used to find a projection of data on a line that maximizes the separation between the class samples.

The classification of CSP filtered signals can be done with a linear classifier. FLD is a simple classifier that has few parameters that need to be tuned. This means there is less over fitting possible. This makes FLD stable and robust.

$$y = w^T x \quad (3.6)$$

It does this by finding the biggest ratio between the scatter between classes and the scatter within classes.

The between class distance can be measured by calculating the distance between the sample means.

$$m_i = \frac{1}{n_i} \sum_{x_i} x \quad (3.7)$$

$$\tilde{m}_i = \frac{1}{n_i} \sum_{x_i} y \quad (3.8)$$

$$= \frac{1}{n_i} \sum_{x_i} w^T x \quad (3.9)$$

The sample scatter is a form of sample variance.

$$\tilde{s}_i^2 = \sum_{y_i} (y - \tilde{m}_i)^2 \quad (3.10)$$

The fischer linear discriminant wants to find a w by maximizing the following formula. Which is the ration mentioned before.

$$J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} \quad (3.11)$$

To solve the equation it needs to be rewritten so that it contains w . If S_w within scatter gets defined as:

$$S_i = \sum_{x_i} (x - m_i)(x - m_i)^t \quad (3.12)$$

$$S_w = S_1 + S_2 \quad (3.13)$$

$$\tilde{s}_i^2 = \sum_{x_i} (w^t x - w^t m_i)^2 \quad (3.14)$$

$$= w^t S_i W \quad (3.15)$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^t S_w W \quad (3.16)$$

$$(3.17)$$

The same thing can be done for the between scatter:

$$S_b = (m_1 - m_2)(m_1 + m_2)^t \quad (3.18)$$

$$(\tilde{m}_1 - \tilde{m}_2)^2 = (w^t m_1 - w^t m_2)^2 \quad (3.19)$$

$$= w^t S_b w \quad (3.20)$$

Now the Fischer LDA criteria formula can be rewritten with w in it:

$$J(w) = \frac{w^t S_b w}{w^t S_w W} \quad (3.21)$$

The maximum of the formula can be by deriving it and equating it to zero. Resulting into the following equation:

$$w = S_W^{-1}(m_1 - m_2) \quad (3.22)$$

3.6 Tools used

This section describes the tools that have been used in the experiments.

3.6.1 Electrodes and Cap

The electrodes used were F1 F2 FC1 FC2 C1 C2 CP1 CP2 P1 P2 F3 F4 FC3 FC4 C3 C4 CP3 CP4 P3 P4 F5 F6 FC5 FC6 C5 C6 CP5 CP6 P5 P6 (Figure 3.5) and were fitted into a cap with the 10 20 system. This systems divides the electrodes evenly over the scalp of the subject. The electrodes are divided with either a distance of 10 or 20 percent of the size of the scalp. Every head size needs to be fitted with a correct size cap.

The bipolar electrodes were applied after scrubbing the skin. They are placed with sticky pads and are filled with EEG conductance gel. This gel makes a better connection but it does evaporate over time which deteriorates the conductance.

The bipolar electrodes were used to detect the eye movements. This could later be used to determine if the cursor control was based on the

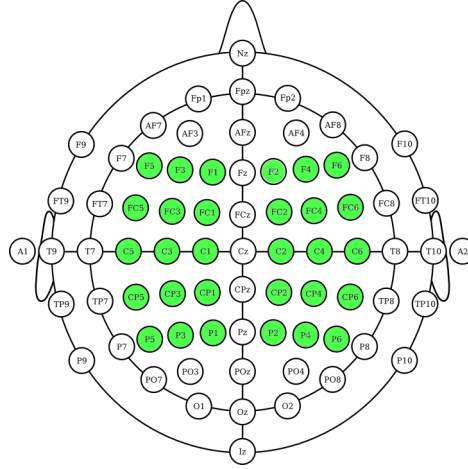


Figure 3.5: The 10-20 EEG Electrode system. The electrodes used in the experiments are marked.

brain activity and not on eye movement. The ground electrode was placed on the chest. And the reference electrodes were placed behind the ears.

The electrodes on the cap were filled with the same EEG electrode gel for better conductance. The conductance of each electrode was measured with an external impedance meter. The impedance was kept below 10 KOhm but the used amplifier is build to work perfect with higher impedance values up to 50 kOhm.

3.6.2 Amplifier

The EEG amplifier takes the signals from the electrodes and amplifies these so they can be used for analysis after digitization. The TMSi refa 5 was used to amplify 30 unipolar electrodes and 2 bipolar electrodes for eye movement detection. Sampling was done with a rate of 250 samples per second. It also has an anti aliasing filtering build in.

3.6.3 Computer & Software

Two computers were used both with a Pentium 4 CPU and enough memory to run Matlab. One computer was used to run the experiments and one to develop the BCI and analyze the collected data. Both were installed with windows XP professional. Matlab was installed along with the BCI2000 program. Also Borland C++ compiler was needed when one of the programs needed to be recompiled.

A 21 inch monitor was used to give the feedback to the subject that sat in front of it with a table distance in between. Both the computers were equipped with two 17 inch monitors.

To create the documentation LaTeX was used. To create the images public domain pictures were used with the GIMP program or Inkspace. Also pictures from papers were used. Whenever a text editor was needed Emacs was used.

3.6.4 Matlab

Matlab is a computer program useful for numerical calculations. It mainly uses matrix calculation and does it very fast in a interactive environment. It is used by many universities and companies for these capabilities. The basic functionality gets extended by many toolboxes that provide easy to use functions for specific domains.

Matlab 2007b was used to do the calculation of the CSP filter and to classify the EEG activity in the online task. With it the signal processing toolbox was used because it gives easy to use filtering and spectrum analysis.

For experimenting with the data, Matlab also proved to be very helpful. It allowed easy analysis of the data to see what sort of information was available. Also it allowed easy off-line experimentation with new classification algorithms.

3.6.5 BCI2000

BCI2000 is a framework for doing BCI research [Schalk et al., 2004]. It has many features that lets the researcher focus more on his research than on the creation of the computer program. The program is a project that is created by the Wadsworth Center and the University of Tübingen.

The framework works with three modules that communicate with each other (Figure 3.6). These modules are Source, Signal processing and Program. The source delivers the data from the brain. For example EEG data.

The signal processing module gets the data from the source module and processes it. Usually this consists of some preprocessing e.g. filtering followed with some feature extraction and classification.

The application module will get a control signal from the processing module and will use this signal to do something useful with it. This is also the module that gives the user feedback.

The following features were used during the research. All the data gets recorded into a single file. The matlab code recorded its own data as well

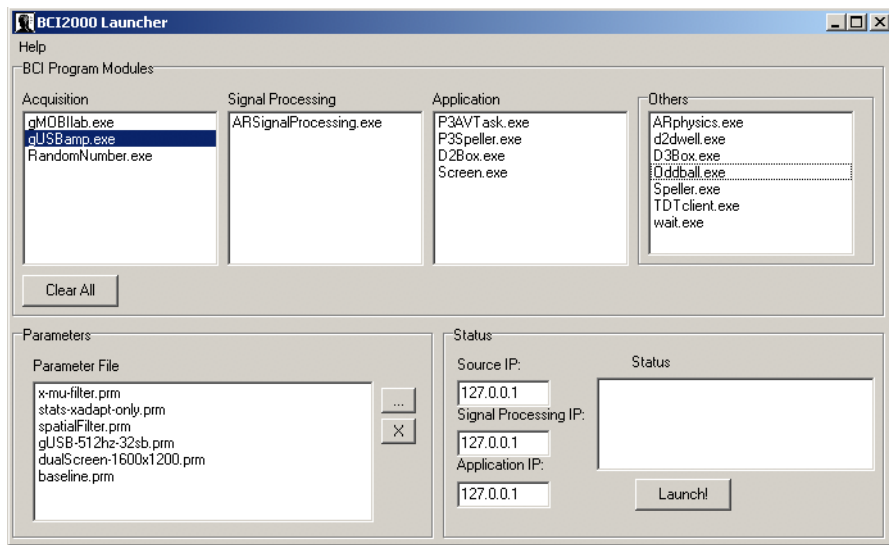


Figure 3.6: The BCI2000 launcher where the three modules can be selected. See text.

but only the part that was of interest. But from time to time data that wasn't recorded was needed and could be retrieved through the BCI2000 file. BCI2000 has a Matlab function that can read all the data collected complete with all the states and information that also gets recorded.

The configuration system of BCI2000 was of great help because it gave the ability to recall the settings and reuse them for a future use. It must be said that it wasn't always easy to get the settings correct to get the entire system working. The documentation doesn't always help in this case because it isn't always very clearly written.

The idea of having three different programs interact with each other has the advantage of re-usability. One could use the same source module (the program that supplies the EEG data) for all experiments. It also makes it easy to shift between the different feedback modules while using the same classification algorithm. The downside (in the way they implemented it) is that individual testing wasn't as easy as it could have been. The modules couldn't be run independent of the operator module. Because the Matlab module was used for Signal processing this drawback could be circumvented by running the Matlab code inside Matlab itself on the recorded data.

The framework is open source for research groups, this allows for small modifications and finding out odd things that aren't in the documentation. Also some examples of modules with source code are available which made it easier to create new modules.

The framework consists of many layers that get formed by many objects that get inherited by other objects. This way of programming makes reuse somewhat easier but it also creates confusion in finding the location of some function that doesn't work like you want it to work.

The communication between modules happens through network connections and goes through the operator module. This operator module takes care of all the time issues and calls all the correct functions inside the other modules. Every module needs to have at least some base functions implemented these then get called in a certain order by the operator module.

BCI2000 isn't perfect but it does save a lot of programming time for it would take more time to get all the basic functionality programmed correctly than to learn to use BCI2000.

3.6.6 Experimentation Room

All the experiments were conducted in a room specially build for EEG research. It has a room where the subject is seated. It is connected to the controller room where the researcher can control the experiment.

The room where the subject is housed in is equipped with a electrical board that allows signals to be entered and retrieved. This is the only place that allows any signals in or out because the room is build like a ferraday cage where all outside electrical disturbance is shielded out. This is necessary because researchers want to be sure only brain wave signals get picked up by the EEG amplifier.

Further the room allows room for a table, a chair and computer equipment. The lights can be regulated from the controller room to create a lighter or darker environment in which the experiments can take place. During the experiments the lights were kept at full strength. The brightness of the screen was adjusted so the subject could maintain a longer fixation on the screen without fatiguing the eyes too much.

The subject room can also be closed to create a soundproof surrounding. This allows the subject to better concentrate on the task. In both rooms a intercom system is installed to allow communication. Also a camera is placed to monitor the subject from within the controller room.

This monitoring is useful to check if the subject is performing correctly and to check if the subject isn't suffering ill effects from the experiments.

The control room has a couple of computers and monitors that control and record the experiment. Only one computer was used for both recording and feedback but with two monitors of which one was transmitted (copied) to the monitor in the subject room.

3.7 Subjects

Two subjects were used to test the TriSpeller system. The first subject participated in an earlier BCI research project on which this thesis continues. The second one was recruited after the first didn't have any more time to continue. From a group of four candidates the one with the highest score was selected. Only one or two subjects were needed because some training is needed to get the basic classification of the imagined hand movement right. And because the interest of the study went out to finding extra signals besides the known signals, less time needed to be spent on training many subjects in basic steering of the system based on these known signals.

The subjects usually underwent two testings of 2 hours every week. Sometimes it was less days in a week because the modification of the system took more time than expected. All the students were paid for their time.

Chapter 4

Hex-O-Spell

“Let the machines learn”

Motto BBCI

The work done by the Berlin BCI (BBCI) group has been the basis for this study. The BBCI has done a lot of work with computer learning algorithms. They work with the motto: let the machines learn. Their effort resulted in a system that lets the user think of certain thoughts that the computer learns to recognize. This results in a short training time for the user.

The thoughts that they were very successful with are thoughts of hand and foot movements. It generally depended on the person what combination was most successful. Many subjects were best classified with hand movement but in some cases a combination of foot and one of the hands was best [Blankertz et al., 2008].

Most of their publications used CSP algorithms to detect the brain patterns in combination with a linear classifier. They also experimented with the adaptation of the bias of the linear classifier. This was mostly done by hand and did make a lot of difference for many subjects [Blankertz et al., 2006a].

The Hex-o-Spell application [Blankertz et al., 2006b] is a spelling device that allows the user to enter letters to form words and sentences. It is one of the best BCI spelling devices and it holds the world record of 7 letters per minute. It uses many ideas of ergonomics and language technology.

4.1 Interface

The interface (Figure 4.1) is made of 6 hexagon figures that form a circle. In the center is an arrow that can be turned clockwise and is used to point

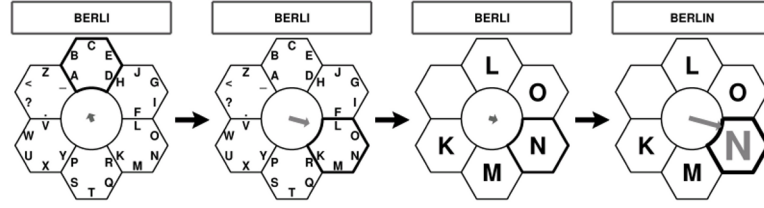


Figure 4.1: The Hex-o-Spell spelling device interface from the Berlin BCI group from [Blankertz et al., 2006b]

towards a hexagon of choice. The arrow can be extended to touch one of the hexagons and with it make the selection.

In each hexagon 5 letters are placed. When one of the hexagons is selected the 5 letters gets distributed over 5 hexagons. Now a letter can be selected.

Because the interface works with a circle it is possible to continue when the selection is missed.

4.2 Language model

The Hex-o-Spell language model used two modified partial predictive-match (PPM) models [Blankertz et al., 2007b]. The first model holds letter probabilities of the Nth letter based on K previous selected letters. BBCI used a K of 2. The second model holds letter probabilities of the Nth letter based on all the previous selected letters. Using relative weighting a final most probable letter gets returned as the result. The model needs to be trained on a large word base. The BBCI used a German newspaper corpus combined with a few German novels.

After the first letter of a word is selected the model gives the most likely letter. With this letter the interface can be rearranged to reduce the time between movement and the selection of the wanted letter. The rearranging of the interface can be done in many ways.

Because the the BBCI group didn't want the user to continually seek for the letter the user wanted to select, they kept the letter groups at the same position at all time. So the letter A to E were always placed at the top, etc. They did arrange the letters within the letter group according to their language model.

The arrow on the other hand was pointed to the most likely letter (group). So when the correct group of letters was in front of the arrow

only the thought for extending the arrow was needed.

4.3 Classification

Using EEG data in combination with CSP filtering and a linear classifier (linear discriminant analysis) results in a classification for steering the interface. The Hex-o-spell uses a binary classifier. It needs about 20 - 30 minutes to calibrate the system. In that time 70 to 200 periods are recorded per thought [Krauledat, 2008]. They used three types of thought; left and right hand movement and foot movement. All combinations were calculated and the best was used [Blankertz et al., 2007a].

4.4 Steering

The steering was done with a signal of continuous speed. The speed can be tailored to the user [Müller et al., 2007]. The output of the classifier gets outputted directly to the interface.

Chapter 5

TriSpeller

The TriSpeller uses most of the best ideas from Hex-o-Spell and improved the system in places that could be engineered better. With the main idea that a third signal could be used.

5.1 Interface

The TriSpeller interface copies the circular hexagon system from Hex-o-Spell because the circular system allows for error correction and puts all the possible letters an equal distance from the starting point. The BBCI systems allowed the arrow to go into only one direction. The system would need to be able to move in both directions. This would give it a speed

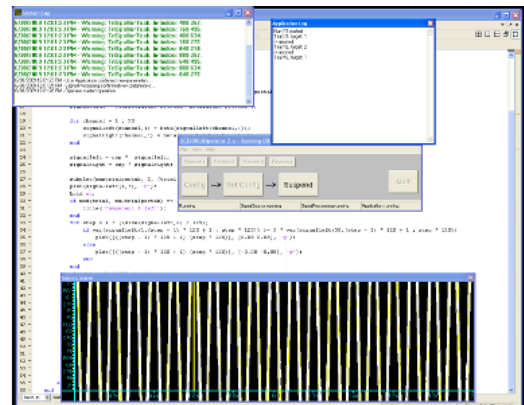
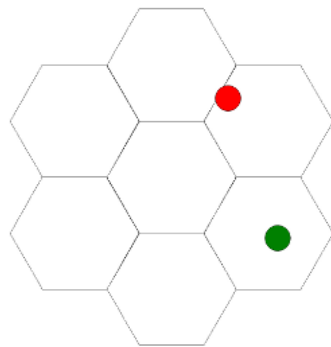


Figure 5.1: The TriSpeller application used in the experiments is based on the Hex-o-Spell. Showing the interface for the calibration task and the BCI2000 on the second screen.

advantage. Now the turning distance to the worst case scenario was reduced by halve. Worst case with the Hex-o-Spell would be going to the hexagon left to the starting hexagon and with TriSpeller this would be the left or right side of the opposite hexagon of the starting hexagon.

5.2 Language model

The language model used is based on a dictionary of a certain language. Based on the letters already selected all the words that fitted these selected letters were collected. After this selection all the letters that followed the already selected letters were collected and it was determined which letter had the highest count and all the found letters were ordered by count.

Based on this ordering the groups or letters were ordered within the interface. The most likely letter or group would be placed on top and the following groups would be placed further away from the top hexagon. And the cursor would always start at the top.

TriSpeller uses letter groups with a size of six letters. This is one more than the Hex-o-Spell interface. By doing this one of the hexagons becomes empty and can be used for something else. The unused hexagon is placed at the bottom furthest away in turning distance from the top starting hexagon. This hexagon now gets filled by the most likely word according to the language model. If the word is correct it is well worth the time to go all the way down to the bottom hexagon because that would save a lot more time.

5.3 Classification

The classification is done with a CSP filter combined with a linear classifier.

The collected data was sorted into left and right hand data in a matlab array. Because the activity of the imagined hand movement is mostly found in the mu frequency region the data was filtered between 8 and 12 hertz. The filtering was done with a band-filter using the `bandfilt` function in Matlab.

Each filtered piece of 3 seconds was then split into sections of 500 msec because the application of the CSP filter would also be done on sections of 500 msec with an overlap of 250 msec. The first and last part of the filtered part were discarded because the filtering causes distortions in these regions.

The CSP was calculated 10 times for part of the sections. The sections were divided into 10 random groups that each were used once as the test group in the K-fold cross validation with K equal to 10.

10 CSP filters were created with a training subset of the data and tested the classification with the testing subset. Doing this ten times gives a good indication if the collected data is good enough to calculate a CSP that is usable in the application.

After the calculations were done the scalp distributions were drawn on the screen for the 3 best performing CSP components that was calculated on the entire data set. For all the 500 msec sections a multiplication with the filter was performed and the variance was calculated and written to a file to be later used as the features for training the linear classifier.

5.4 Steering

The control signal drives the arrow in the application. The control signal is based on the classification results. When passing only the classification result the system could be wrong because the classification isn't perfect when it isn't 100 % correct. When the system is wrong the arrow would jump into the wrong direction.

Because this gives a restless arrow movement it was decided to take the results from the past and give some sort of average to smooth the movement. This is allowed because the intended direction is usually towards one direction after which it is stopped at the correct position in the interface. Because the oldest classification was of less important than the newest a linear formula for the steering signal was devised:

$$outSignal = class_1 * \frac{20}{10} + class_2 * \frac{19}{10} + \dots + class_{20} * \frac{1}{10} \quad (5.1)$$

5.5 Gray area

CSP is a technique that ultimately only can distinct two states. When you combine this with a linear classifier you always will get a classification even when the linear classifier isn't very sure. This in effect means that when the user of the system doesn't think of either the right or left hand movement it will give a classification. This is a big drawback of the CSP linear classification combination.

To make the arrow stop when the user isn't thinking of any hand movement a gray area was created in the classification. This is done automatically using the separability value that is returned by the FLD function that comes with Matlab.

The values that come from the `linclass` Matlab function are used to determine the gray area. The `linclass` function gives a classification but also a value coming from the model. This value is positive for one class and negative for the other. The more distance there is between the centers of the class values the better the classification can be made.

The median of each of the class values and the distance between them was calculated. This region was used as the gray area. If the class value is bigger than the gray area value or smaller than the negative gray area value the left and right classification was given of. This results with almost halve of each of the classifications being ignored.

5.6 Bias adaptation

The users of the system complained that it is sometimes impossible to turn to one side while the other side is easy. This is possible because the classification is done with a linear classifier which can have a slight preference for one side over the other. Especially when the two groups differ very little. This preference can be adjusted by a bias value that can shift the center between the two groups.

This adjusting can be done by hand but also automatically with small steps whenever a wrong selection was made (when the longest road to the target was taken). To decide what the small steps had to be the separability value was used. Through experimentations on data it was found that the separability value divided by 20 worked good as the step value.

$$Model.b = Model.b - Model.separab/20$$

5.7 Testing

During the building process many small tests were performed on two subjects. The subjects needed to move to a target in the interface and when reached perform different tasks depending on the test.

5.7.1 Results

The correctness scores did grow gradually after training. After only a few training sessions the scores were in the high eighties low nineties which were very high when looking at other research groups. Also the control of the arrow was very good. Figure 5.2 shows a CSP plot belonging to a high

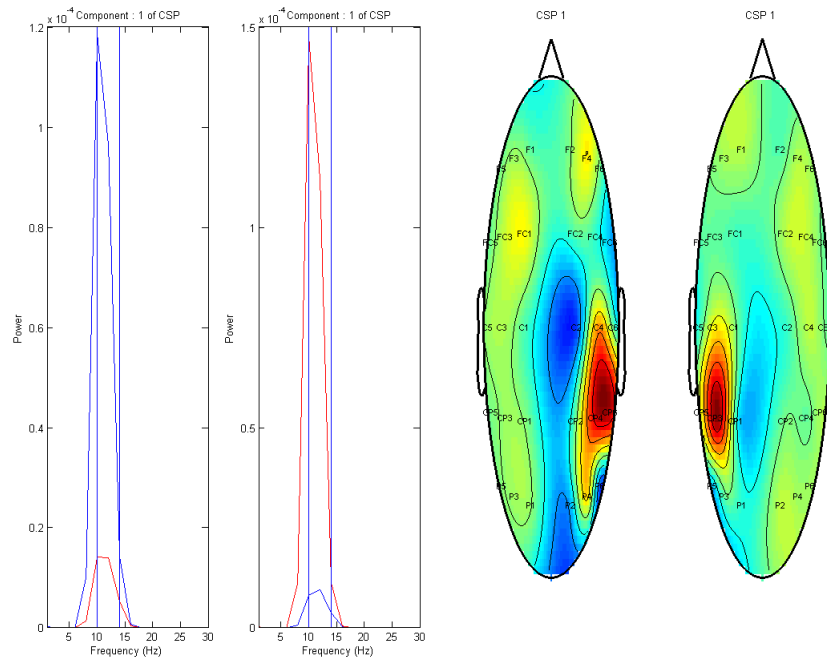


Figure 5.2: A figure from CSP calculated for the hand movements in the Mu frequency region.

score in the low nineties. If the user needed to move the cursor there was no hesitation in the movement.

The calculation of the steering signal resulted in a good control but also a fast change in the direction of the arrow but not so much to create a restless arrow when an error was made in the classification.

Activating the gray area function did give three states that the arrow could be in. Left, right and no movement but it did make it harder to get the arrow to move at all and wasn't as predictable as before. It was possible to stop but it was hard to make it stop on command.

When the automatic bias adaptation was used it became easier to create a more balanced control. But this was only necessary when control was poorly. One of the subjects had such high scores and control that the bias adaptation couldn't be even tested.

The adaptation is also only possible when the target is known. This technique isn't possible when the user is free spelling. Some knowledge is needed about which target was to be selected. This is the whole idea behind BCI so the best solution behind the bias problem is to have good control.

5.7.2 Discussion

Further research could be done in making a better gray area. Also more training on the subject part could result in better controlling the starting and stopping of the cursor. It was decided that this method wouldn't result in the success that was sought after. A third signal needed to be found to create a method for stopping the cursor.

Chapter 6

Pilot experiment

“Standing on the shoulders of giants.”

Bernard of Chartres 1159

Beta rebound is a brain signal that is found when a movement is stopped but also when a switch in movement is created. This signal can be used to detect when a movement is stopped and used as a stop signal for the task within the interface.

6.1 Basic paradigm

All the experiments have the same basic paradigm. The interface was kept the same as is described in the TriSpeller chapter. All the experiments started out with a classification run followed by a training run all using the same basic task. After a classification of above 85% was reached the user could perform the user training run in which the user could train in using the system. After good steering capabilities were shown the real experiment could start.

6.1.1 Task

The basic task is to think of movement of the right or left hand and arm. No actual movement was allowed and the subject was instructed to remain relaxed during the experiment. The jaws needed to be loose.

The user needed to imagine two hand movements, one with each hand. This had to be a movement that could be maintained for a short period. After some experimentation a large circular movement with almost the complete arm involved was chosen as the imagined movement. The right hand

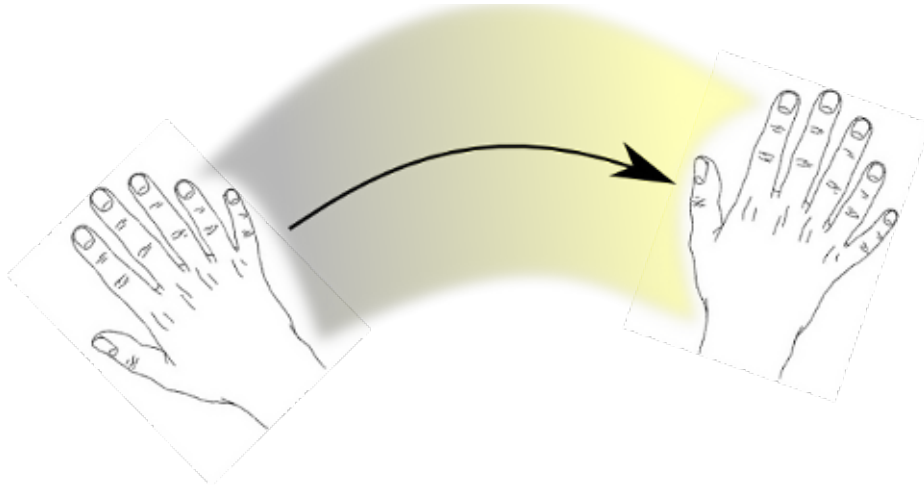


Figure 6.1: Representation of the right hand movement that needed to be imagined by the subject.

made a circular movement clockwise and the left hand went anti clockwise (Figure 6.1).

6.1.2 Classification run

Before the real experiment could be performed the BCI needed to be trained. This training of the system is called the classification run. During this run brain activity is collected that gets used to calculate a CSP filter and train the linear classifier.

The amount of sections that was recorded per hand is 20. The duration of the sections is 3 seconds. The subject was shown the interface with a left or right target. This target is used as a cue for the subject to think of the left or right hand movement. The subject wasn't shown feedback during this run.

6.1.3 User training run

When the classification run is finished with a high score (above 85%) then the subject gets to do the same task as in the classification run but now with feedback. The feedback is generated by the BCI and allows the subject to train with the system.

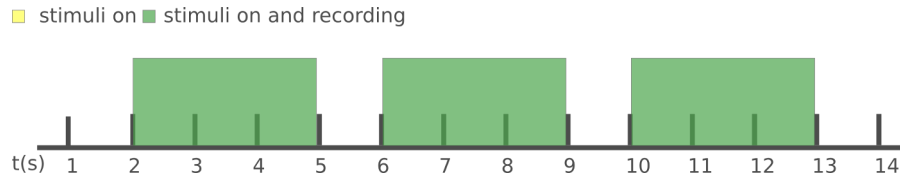


Figure 6.2: Stimuli presentation for the betarebound experiment

6.1.4 Experiment run

When the subject and experimenter found that the subject had a good control over the cursor, the experimentation run could be performed. Again the user gets a task with feedback. A task was devised that demanded a fast switch of the hands. The stimuli wasn't random, it always switched from one hand to the other.

Stimuli

The stimuli was presented for 3 seconds with a second between the stimuli (Figure 6.2). Both stimuli and pause periods were recorded separately in a Matlab matrix.

It was hypothesized that the beta rebound would be in the second between the stimuli. This section was filtered around the beta frequency range.

6.1.5 Results

Based on previous research it was expected that the beta rebound was within the 1 second space between the stimuli but it wasn't found. It was decided to look a little further in the data because some extra beta activity was found within the beginning of the movement thoughts sections (see figure 6.3). This showed that the beta rebound occurred later and was somewhat slower than in previous experiments.

A perfect 100% or a high 99% score was found when the CSP was calculated on the beta component. In the calculation 280 segments of 500 msec with overlap of 250 msec was used for each rebound side.

6.1.6 Discussion

Now that a filter was calculated the detection of a beta component could be performed. Because of the working of the linear classifier it always gives a result whether there was a beta rebound or not. So a combination of both

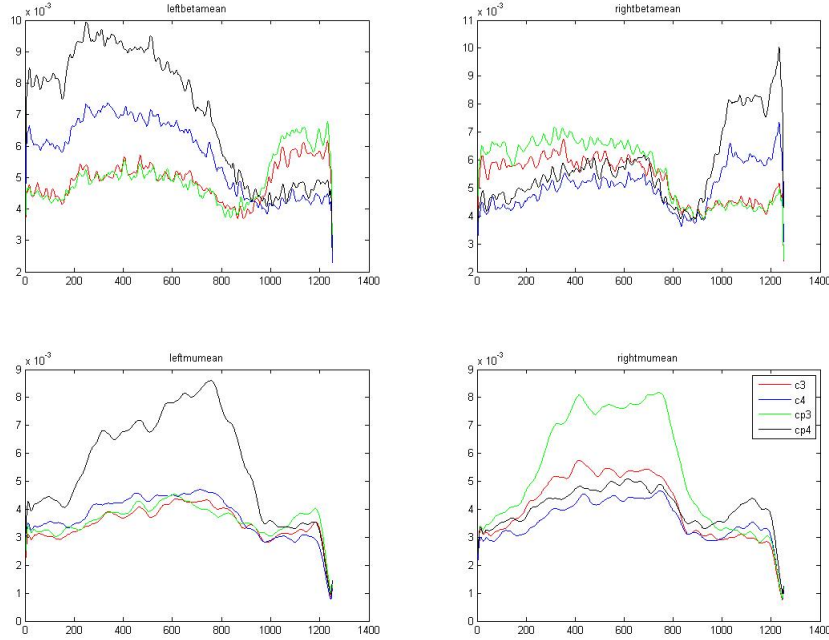


Figure 6.3: Powerplot: 0 represents the start of the stimuli recording which stops at 750 (3sec) and the next one starts at 1000. The top part are filtered at the beta range and the bottom part in the mu range. It shows that beta activity starts after the 1000 sample point

the left right signal and rebound signal needs to be used to create a system that can give a correct selection signal.

6.2 Selecting letters

After the success in finding the beta rebound CSP components a beta rebound detection algorithm was created with high accuracy results (Figure 6.4).

The algorithm takes the knowledge of the route the arrow is taking. With this knowledge it knows what classification to listen to. Next the algorithm looks at the variance of the best CSP components and determines if one is three times greater than the other. When this is the case it will give a detection signal.

With this algorithm the final application could be tested. The only

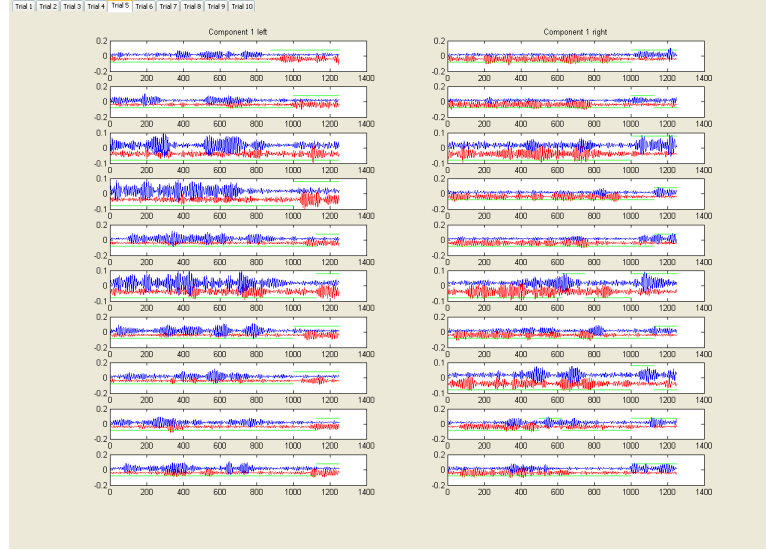


Figure 6.4: Beta detection shown for a few of the trials. The green line shows when a detection is made by the algorithm. When it is up a beta burst is detected. This should happen around the 1000 sample point.

trouble was that only one week was left for experimentation.

6.2.1 Experimental task

The subject needed to guide the cursor to the target in the interface. When the target was reached a switch to the other hand movement needed to be made.

6.2.2 Results

Good results were gotten in detecting the beta component when the control was good but when control was lost the beta component also was lost. This is somewhat logical because these signals are highly dependent of each other. Two experimentations on different days were performed but the control was so much worse as compared to the experiments in the testing phase of the main system.

But even when two more weeks of experimentation could have been done problems were expected with probable lag in the detection of the beta component that needed to be tweaked.

Chapter 7

Discussion

This thesis has resulted in a better platform to continue the BCI research at the university of Groningen. The platform allows the user and researcher to have a good working BCI within 10 minutes. Also some little features were developed that could be useful. The pilot study showed that the beta rebound is a promising signal that could be used as the third signal. Further study needs to be done to confirm this.

The beta rebound is found better when a direct switch is made between two imagined movements. This could suggest that it is stronger when fast changes occur in movement where one movement needs to be suppressed more to make the other movement work better. These ideas warrant further studying.

In the developing of the system some work was done on the bias adaptation. Other groups also showed that there is something to be gained when the bias is tweaked to a ideal value [Blankertz et al., 2006a]. The system tests did show that when classification is of high correctness the need for bias tweaking isn't necessary. It would be interesting to determine why the classification score is higher when it is. What is different in the brain activity between those classifications. Is this brain specific, can it be trained, is there a way to further improve it?

Because the system has to calculate the classification with a certain amount of data and because there is some processing time involved in the human brain there will be some delay in the closed loop that needs to be dealt with. What kind of techniques need to be used to create the correct amount of delay. Does the brain learn to adjust to the delay over time?

The speed of the arrow movement is determined on the previous classifications. The way it is calculated could be made to work better than it does.

The drawback of CSP with a linear classifier is that it only uses two

states. It is either class one or class two. When working with this constraint it is possible to create some working system. It would be more natural to have a state in which it is sure that neither state (left right) is reached. This makes it easier to test the classification but also to maybe add more states more easily.

In this thesis only one classification technique was tested. Other techniques have been demonstrated to work as well or even better than the CSP technique. It would be interesting to compare different techniques against our CSP implementation. This could be easily done by replacing the matlab module and perform the calculation on collected data. After this online testing could be done to see if the theoretical improvement is also true in practice.

Also the interface could be changed to determine if other ways of presenting the feedback could make for a better and faster system.

Using a different language model that uses some sort of statistical basis would likely increase the speed of the system even further.

Most studies have been done on healthy subjects and this of course reasonable because communication with locked in patients is hard and this is difficult when explorative research is done. The other reason is that real locked in patients don't have a long lifespan and with it are very small in numbers per country. But when a system works perfectly on healthy persons it is not said that it works well on locked in patients. Testing should be done on locked in patients to be sure that the found results also apply for this group.

The TriSpeller system shows that signals in longer brain activity can be detected with high accuracy. Most other systems test their subjects with very short distance targets. These don't need the subjects to maintain their brain activity for very long. When these distances are short many errors can disappear because they only show up in longer intervals.

All in all the ideas and findings in this thesis warrant further study because using all the available information within the BCI combined with beta rebound could seriously improve the lives of locked in patients and enhance the experience of healthy persons.

Bibliography

- [Birbaumer et al., 1999] Birbaumer, N., Ghanayim, N., Hinterberger, T., Iverson, I., Kotchoubey, B., Kubler, A., Perelmutter, J., Taub, E., and Flor, H. (1999). A spelling device for the paralysed. *Nature*, 398(6725):297–298.
- [Blankertz et al., 2007a] Blankertz, B., Dornhege, G., Krauledat, M., Müller, K., and Curio, G. (2007a). The non-invasive Berlin Brain–Computer Interface: Fast acquisition of effective performance in untrained subjects. *Neuroimage*, 37(2):539–550.
- [Blankertz et al., 2006a] Blankertz, B., Dornhege, G., Krauledat, M., Müller, K., Kunzmann, V., Losch, F., and Curio, G. (2006a). The Berlin Brain-Computer Interface: EEG-based communication without subject training. *hand*, 3:C4.
- [Blankertz et al., 2006b] Blankertz, B., Dornhege, G., Krauledat, M., Williamson, J., and Murray-Smith, R. (2006b). The Berlin Brain-Computer Interface presents the novel mental typewriter Hex-o-Spell. In *Proceedings of the 3rd International Brain-Computer Interface Workshop and Training Course 2006*.
- [Blankertz et al., 2008] Blankertz, B., Krauledat, F., Dornhege, M., Curio, G., Müller, G., et al. (2008). The Berlin Brain–Computer Interface: Accurate Performance From First-Session in BCI-Naïve Subjects. *IEEE Transactions on Biomedical Engineering*, 55(10).
- [Blankertz et al., 2007b] Blankertz, B., Krauledat, M., Dornhege, G., Williamson, J., Murray-Smith, R., and Müller, K. (2007b). A note on brain actuated spelling with the Berlin Brain-Computer Interface. *Lecture Notes in Computer Science*, 4555:759.
- [Cincotti, 2007] Cincotti, F. (2007). Vibrotactile Feedback for Brain-Computer Interface Operation. *Computational Intelligence and Neuroscience*, 2007:1–12.

- [Cohen, 1995] Cohen, P. R. (1995). *Emperical methods for artificial intelligence*. The MIT Press.
- [Coyle et al., 2004] Coyle, S., Ward, T., Markham, C., and McDarby, G. (2004). On the suitability of near-infrared (NIR) systems for next-generation brain-computer interfaces. *Physiol. Meas.*, 25:815–822.
- [Kalat, 1998] Kalat, J. W. (1998). *Biological psychology 6th ed.* Brooks/-Cole Publishing Company.
- [Krauledat, 2008] Krauledat, J. (2008). *Analysis of Nonstationarities in EEG Signals for Improving Brain-Computer Interface Performance*. PhD thesis, Universit
”atsbibliothek.
- [Leuthardt et al., 2004] Leuthardt, E., Schalk, G., Wolpaw, J., Ojemann, J., and Moran, D. (2004). A brain computer interface using electrocorticographic signals in humans. *Journal of Neural Engineering*, 1(2):63–71.
- [Lotte et al., 2007] Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., and Arnaldi, B. (2007). A review of classification algorithms for eeg-based braincomputer interfaces. *J. Neural Eng.*, 4(2):R1+.
- [McFarland et al., 2000] McFarland, D., Miner, L., Vaughan, T., and Wolpaw, J. (2000). Mu and beta rhythm topographies during motor imagery and actual movements. *Brain Topography*, 12(3):177–186.
- [Mellinger et al., 2007] Mellinger, J., Schalk, G., Braun, C., Preissl, H., Rosenstiel, W., Birbaumer, N., and Kübler, A. (2007). An MEG-based brain-computer interface (BCI). *Neuroimage*, 36(3):581–593.
- [Middendorf et al., 2000] Middendorf, M., McMillan, G., Calhoun, G., and Jones, K. (2000). Brain-computer interfaces based on the steady-state visual-evokedresponse. *Rehabilitation Engineering, IEEE Transactions on [see also IEEE Trans. on Neural Systems and Rehabilitation]*, 8(2):211–214.
- [Müller et al., 2007] Müller, K., Tangermann, M., Dornhege, G., Krauledat, M., Curio, G., and Blankertz, B. (2007). Machine learning for real-time single-trial EEG-analysis: from brain-computer interfacing to mental state monitoring. *Journal of neuroscience methods*, 167(1):82–90.

- [Nijboer et al., 2008] Nijboer, F., Furdea, A., Gunst, I., Mellinger, J., McFarland, D., Birbaumer, N., and Kübler, A. (2008). An auditory brain-computer interface (BCI). *Journal of Neuroscience Methods*, 167(1):43–50.
- [Pascual-Marqui et al., 1994] Pascual-Marqui, R., Michel, C., and Lehmann, D. (1994). Low resolution electromagnetic tomography: a new method for localizing electrical activity in the brain. *International Journal of Psychophysiology*, 18(1):49–65.
- [Pfurtscheller et al., 2005] Pfurtscheller, G., Neuper, C., Brunner, C., and da Silva, F. (2005). Beta rebound after different types of motor imagery in man. *Neuroscience Letters*, 378(3):156–159.
- [Pfurtscheller and Solis-Escalante, 2009] Pfurtscheller, G. and Solis-Escalante, T. (2009). Could the beta rebound in the EEG be suitable to realize a brain switch? *Clinical Neurophysiology*, 120(1):24–29.
- [Piccione et al., 2006] Piccione, F., Giorgi, F., Tonin, P., Priftis, K., Giove, S., Silvoni, S., Palmas, G., and Beverina, F. (2006). P300-based brain computer interface: Reliability and performance in healthy and paralysed participants. *Clinical Neurophysiology*, 117(3):531–537.
- [Schalk et al., 2004] Schalk, G., Mcfarland, D. J., Hinterberger, T., Birbaumer, N., and Wolpaw, J. R. (2004). Bci2000: a general-purpose brain-computer interface (bci) system. *Biomedical Engineering, IEEE Transactions on*, 51(6):1034–1043.
- [Sellers and Donchin, 2006] Sellers, E. and Donchin, E. (2006). A P300-based brain-computer interface: Initial tests by ALS patients. *Clinical Neurophysiology*, 117(3):538–548.
- [Weiskopf et al., 2004] Weiskopf, N., Mathiak, K., Bock, S., Scharnowski, F., Veit, R., Grodd, W., Goebel, R., and Birbaumer, N. (2004). Principles of a Brain-Computer Interface (BCI) Based on Real-Time Functional Magnetic Resonance Imaging (fMRI). *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, 51(6).
- [Wolpaw et al., 2002] Wolpaw, J., Birbaumer, N., McFarland, D., Pfurtscheller, G., and Vaughan, T. (2002). Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791.

- [Wolpaw and McFarland, 2004] Wolpaw, J. R. and McFarland, D. J. (2004). Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proc Natl Acad Sci U S A*, 101(51):17849–17854.

Appendix A

Matlab Code

A.1 bci Initialize.m

```
function bci_Initialize( in_signal_dims, out_signal_dims )

global bci_Parameters bci_States gray_area;

% We use a global variable for our filter configuration as well.

global nsamp Plot CSPFilterMatrix betacsp TrainingFeatures ←
    TrainingGroup Patterns BlockSize HistoryBlocks sr outfile ←
    lpfreq hpfreq signal FilterA FilterB FilterBetaA FilterBetaB ←
    Model Bias Calibrate CalibrationSignal ntrials prevtrial ←
    filt_signal prevtarget;

signal=[];
CSPFilterMatrix = str2double( bci_Parameters.CSPFilterMatrix );
TrainingFeatures= str2double( bci_Parameters.TrainingFeatures );
TrainingGroup   = str2double( bci_Parameters.TrainingGroup );
Patterns        = str2double( bci_Parameters.UsePatterns);
BlockSize       = str2double( bci_Parameters.SampleBlockSize);
HistoryBlocks   = str2double( bci_Parameters.HistoryBlocks);
Bias            = str2double( bci_Parameters.Bias);
Calibrate       = str2double( bci_Parameters.Calibrate);
Plot            = str2double( bci_Parameters.Plot);
nt              = str2double( bci_Parameters.NumberOfTrials);
sr              = str2double( bci_Parameters.SamplingRate);

Calibrationsignal{2,nt}=[];
ntrials=[0 0];
prevtrial=0;
prevtarget=-1;

data.X=TrainingFeatures(:,[1:Patterns end-(Patterns-1):end])';
data.y=TrainingGroup';
Model=fld(data);
Bias= Model.b;

if (str2double(bci_Parameters.GrayArea))
```

```

    [class dfce] = linclass(data.X, Model);
    gray_area = (median(dfce(TrainingGroup == 1)) - median(dfce(←
        TrainingGroup == 2))) / 2;
else
    gray_area = 0;
end

if (str2double(bci_Parameters.EditBias))
    Bias = str2double( bci_Parameters.Bias);
else
    bci_Parameters.Bias {1,1}= num2str( Bias );
end
nsamp=0;

Model.b=Bias;

lpfreq=str2double(bci_Parameters.lpfreq );
hpfreq=str2double(bci_Parameters.hpfreq );
filters=sort([lpfreq hpfreq]);
save 'c:\test.mat' filters;
[FilterB,FilterA]= bandfilt(sr,filters(1)-2,filters(1),filters(2),←
    filters(2)+2,1,30);
% beta component toevoeging: filters en betacsp
[FilterBetaB,FilterBetaA] = bandfilt(sr,15 - 2, 15, 30, 30 + 2, 1, ←
    30);
load( 'c:\betacsp.mat' );

```

A.2 bci Process.m

```

function out_signal = bci_Process( in_signal )
% Parameters and states are global variables.
global bci_Parameters bci_States;

% We use a global variable to store our filter's configuration.
global betacsp CSPFilterMatrix TrainingFeatures TrainingGroup ←
    Patterns BlockSize HistoryBlocks prevtarget outfile lpfreq ←
    hpfreq filters FilterA FilterB FilterBetaA FilterBetaB sr Model←
    Calibrate CalibrationSignal ntrials prevtrial;
global signal memclass filt_signal modelupdated gray_area;
f=fopen('c:\testa.txt', 'a');
if Calibrate
    Trial=bci_States.TrialNumber;
    Target=bci_States.TargetCode;
    out_signal=(3-(Target*2));
    if (bci_States.Feedback==1) % Stimulus on screen
%         fprintf(f, 'Trial: %i (%i) \n', Trial, prevtrial);
        if (Trial~=prevtrial) % trial has ended
%             fprintf(f, 'Trial: %i (%i) - ', Trial, prevtrial);
%             fprintf(f, 'Target: %i (%i) - ', Target, prevtarget);
%             fprintf(f, 'Signal size: (%i %i)\n', size(signal));
            if prevtarget>0
                ntrials(prevtarget)=ntrials(prevtarget)+1;

```



```

%          fprintf(f, '#Target: %i (Trials:%i %i)\n', ←
prevtarget, ntrials);
          CalibrationSignal{prevtarget,ntrials(prevtarget)}= ←
          signal;
      end
      signal=[];
      prevtarget=Target;
      prevtrial=Trial;
  end
  signal=[signal in_signal];
end
else % On-Line Processing of the Data

  signal=[signal in_signal];
  if size(bci_Parameters) == 0
      bci_States.Feedback=1;
  end
  if bci_States.Feedback==1
      modelupdated = 0;
      if (size(signal,2) >= HistoryBlocks*BlockSize)
          while (size(signal,2) > HistoryBlocks*BlockSize)
              signal=signal(:,BlockSize+1:end);
          end
          for channel=1:size(signal)
              filt_signal(channel,1:size(signal,2)) = filtfilt(←
                  FilterB, FilterA, signal(channel,:));
              beta_filt_signal(channel,1:size(signal,2)) = ←
                  filtfilt(FilterBetaB, FilterBetaA, signal(←
                      channel,:));
              %refreshdata;
          end

          filt_signal=(CSPFilterMatrix * filt_signal);
          beta_filt_signal =( betacsp * beta_filt_signal);

          for in=1:Patterns
              feature(in)= var(filt_signal(in,end - 6 * BlockSize←
                  :end - BlockSize));
              feature(2*Patterns + 1 - in)= var(filt_signal(size(←
                  signal,1) + 1 - in,end - 6 * BlockSize:end - ←
                  BlockSize));
          end

          totvar= sum(feature(1:2*Patterns));

          for in=1:2*Patterns
              logf(in)= log(feature(in)/totvar);
          end

          [class dfce] = linclass(logf',Model);
          class = zeros(1,size(class,2));
          class(find(dfce > gray_area)) = -1;
          class(find(dfce < -gray_area)) = 1;

          %prev=median(memclass);

          memclass=[class memclass];

          if size(memclass,2)>20

```

```

        memclass=memclass(1,1:20);
    end
    %if prev ~= class memclass=class; end
    %out_signal=[sum(memclass) sum(memclass)];

    %out_signal=[abs(sum(memclass))*class abs(sum(memclass)←
    )*class ];
    %out_signal=sum(memclass); % moet betere functie voor ←
    komen

    out_signal = 0;
    for i=1:size(memclass,2)
        out_signal = out_signal + memclass(i) * ((20 - i) /←
        10);
    end

    % betacomponent detectie geeft 55 terug als stopsignaal←
    via out_signal.
    if var(beta_filt_signal(1,end - 6 * BlockSize:end - ←
    BlockSize)) >= 3 * var(beta_filt_signal(30,end - 6 ←
    * BlockSize:end - BlockSize)) && out_signal > 0
        out_signal = 55;
    end
    if var(beta_filt_signal(30,end - 6 * BlockSize:end - ←
    BlockSize)) >= 3 * var(beta_filt_signal(1,end - 6 *←
    BlockSize:end - BlockSize)) && out_signal < 0
        out_signal = 55;
    end

    if size(bci_Parameters) ==0
        hold on;
        subplot(2,1,prevtarget);
        plot(prevtrial,out_signal);
    end
end
%out_signal=class;
else %we're in the ITI
    if size(modelupdated) == 0
        modelupdated = 0;
    end
    if (bci_Parameters.AutomaticBiasAdaptation{1} == '1' && ←
    modelupdated == 0)
        out_signal = 0;
        for i=1:size(memclass,2)
            out_signal = out_signal + memclass(i) * ((20 - i) /←
            10);
        end
        fprintf(f, 'Targetcode: %d \n', bci_States.TargetCode);
        fprintf(f, 'out_signal: %d \n', Model.separab);
        fprintf(f, 'Bias: %d \n', Model.b);
        if (bci_States.TargetCode == 3 || bci_States.TargetCode←
        == 2) && out_signal < 0
            Model.b = Model.b - Model.separab / 20;
        end
        if (bci_States.TargetCode == 5 || bci_States.TargetCode←
        == 6) && out_signal > 0
            Model.b = Model.b + Model.separab / 20;
        end
    end
    modelupdated = 1;

```

```

        end
        out_signal=0;
        memclass=[];
    end

end% On-Line Processing of the Data
fclose(f);

```

A.3 bci StopRun.m

```

function bci_StopRun

% Filter stop run demo
%
% Perform parameter updates at the end of a run.

% BCI2000 filter interface for Matlab
% juergen.mellinger@uni-tuebingen.de, 2005

% Parameters and states are global variables.
global bci_Parameters bci_States gray_area betarange;
global Plot CalibrationSignal Filter Calibrate BlockSize ←
    HistoryBlocks Patterns FilterA FilterB ntrials signal ←
    prevtarget;

if size(bci_Parameters) ==0
    lpfreq=8;
    hpfreq=12;
    if betarange == 1
        lpfreq = 15;
        hpfreq = 30;
    end
    sr=250;
    Calibrate=1;
    BlockSize=25;
    Plot = 1;
    HistoryBlocks=8;
%    HistoryBlocks=6;
    Patterns=2;
    filters=sort([lpfreq hpfreq]);
    [FilterB,FilterA]= bandfilt(sr,filters(1)-2,filters(1),filters←
        (2),filters(2)+2,0.1,30);
    if(Plot)
        figure;
    end
    %labels={'f1'; 'fc1'; 'f2'; 'fc2'; 'a1'; 'a2'; 'eog1'; 'eog2'; '←
        emg1'; 'emg2'; 'aux1'; 'aux2'; 'aux3'; 'aux4'; 'test'};
    labels={'f1'; 'f2'; 'fc1'; 'fc2'; 'c1'; 'c2'; 'cp1'; 'cp2'; 'p1←
        '; ...
        'p2'; 'f3'; 'f4'; 'fc3'; 'fc4'; 'c3'; 'c4'; 'cp3'; 'f5'; '←
        p3'; ...
        'p4'; 'cp4'; 'f6'; 'fc5'; 'fc6'; 'c5'; 'c6'; 'cp5'; 'cp6'; ←
        'p5'; 'p6'};% ... F5 en CP4 op versterker verkeerd ←

```

```

        aangesloten nu met labels aangepast
% 'a1'; 'a2'; 'eog1'; 'eog2'; 'emg1'; 'emg2'; 'aux1'; 'aux2'; '←
    aux3'; 'aux4'; 'test'};

load('c:\cs.mat')
else
    lpfreq=str2double(bci_Parameters.lpfreq);
    hpfreq=str2double(bci_Parameters.hpfreq);
    sr=str2double(bci_Parameters.SamplingRate);
    labels=bci_Parameters.ChannelNames;
    if size(labels)==0
        labels={'f1'; 'f2'; 'fc1'; 'fc2'; 'c1'; 'c2'; 'cp1'; 'cp2'; ←
            'p1'; ...
            'p2'; 'f3'; 'f4'; 'fc3'; 'fc4'; 'c3'; 'c4'; 'cp3'; 'cp4'; ←
            'p3'; ...
            'p4'; 'f5'; 'f6'; 'fc5'; 'fc6'; 'c5'; 'c6'; 'cp5'; 'cp6'; ←
            'p5'; 'p6'; ...
            'a1'; 'a2'; 'eog1'; 'eog2'; 'emg1'; 'emg2'; 'aux1'; '←
            aux2'; 'aux3'; 'aux4'; 'test'};
        nchan=size(CalibrationSignal{1,1},1);
        labels=labels(1:nchan);
    end
    if Calibrate
        if size(bci_Parameters) ~=0
            %% Save data from the last trial
            ntrials(prevtarget)=ntrials(prevtarget)+1;
            CalibrationSignal{prevtarget,ntrials(prevtarget)}= ←
                signal;
            signal=[];
            prevtarget = -1;
            clear filt_signal;
        end
    end
    save('c:\cs.mat','CalibrationSignal')
end

if Calibrate

    %filterb=sort([lpfreq hpfreq]);

    m_num= 3; % maximaal aantal CSP componenten
    nfreq= 5; % must be an odd number
    midfreq = ceil(nfreq/2);
    totcor= zeros(nfreq,m_num);
    gray_area = zeros(nfreq,1);

    singPro('Training the Classifier...');
    tic

    for ifreq=1:nfreq

        filters=sort([lpfreq hpfreq]);
        filters(1)= filters(1) + ifreq - midfreq;
        filters(2)= filters(2) + ifreq - midfreq;

        [FilterB,FilterA]= bandfilt(sr,filters(1)-2,filters(1),←
            filters(2),filters(2)+2,0.1,30);

        nchan=size(CalibrationSignal{1,1},1);

```

```

nsamp=size(CalibrationSignal{1,1},2);
ncat=size(CalibrationSignal,1);
maxtrial=size(CalibrationSignal,2);
ntrial=zeros(1,ncat);

eeg=zeros(nchan,nsamp,maxtrial,ncat);
eeg_epoch = zeros(nchan,floor(sr/2),nsamp*2/sr, ncat);
new_ntrial= 0 * (1:ncat);
for hand=1:ncat
    for trial=1:maxtrial
        if (size(CalibrationSignal{hand,trial})) ~= 0
            ntrial(hand)=ntrial(hand)+1;
            for channel=1:nchan
                eeg(channel,:,trial,hand) = filtfilt(←
                    FilterB, FilterA, CalibrationSignal{←
                        hand,trial}(channel,:));
            end
            for i=1:1000
                % beg= floor(sr/2 + (i-1) * sr/4);
                % beg= floor(sr/4 + (i-1) * sr/4);
                % beg= floor(3 * sr/2 - 3 * sr/10 + (i-1) * ←
                    sr/10);
                % fin= floor(beg + sr/2 - sr/10 - 1);
                % fin= floor(beg + sr/2 - 1);
                % ns2= floor(fin-beg + 1);
                % if fin <= (nsamp - sr/10)
                %     if i <= 1
                %         new_ntrial(hand)= new_ntrial(hand) + 1;
                %         eeg_epoch(:,1:ns2,new_ntrial(hand),hand)←
                    %         = eeg(:,beg:fin,trial,hand);
                %     else
                %         break;
                %     end
            end
        end
    end
end

clear eeg
eeg= eeg_epoch;
clear eeg_epoch
ntrial= new_ntrial;

% one more correction for mean
for hand=1:ncat
    for j=1:ntrial(hand)
        for i=1:nchan
            eeg(i,:,j,hand)= eeg(i,:,j,hand) - mean(eeg(i←
                ,:,j,hand));
        end
    end
end

% now on to the 10 K cross-validation
eeg_t = eeg;

step_10 = zeros(ncat);
for i=1:ncat
    step_10(i)= ntrial(i)/10;
end

```

```

        p(i,1:ntrial(i))= randsample(ntrial(i),ntrial(i));
    end

    a_cov = zeros(nchan, nchan, ncat);
    min_gray = 0;
    max_gray = 0;
    for krun=1:10
        selection = ones(ntrial(1),ncat);
        for i=1:ncat
            beg(i)= fix((krun-1) * step_10(i) + 1);
            fin(i)= fix(krun * step_10(i));
            x_test(i)= fin(i) - beg(i) + 1;
            x_train(i)= ntrial(i) - x_test(i);
            for j=beg(i):fin(i)
                selection(p(i,j),i) = 0;
            end
        end
        a_cov = 0 * a_cov;
        for hand = 1:ncat
            for trial = 1:ntrial(hand)
                if selection(trial,hand) == 1
                    dumx(:, :) = eeg(:, :, trial, hand);
                    y = dumx * dumx';
                    y = y / trace(y);
                    a_cov(:, :, hand) = a_cov(:, :, hand) + y;
                end
            end
            a_cov(:, :, hand) = a_cov(:, :, hand)/sum(selection(1:↵
                ntrial(hand),hand));
        end
    end

    [W,D]= eig(a_cov(:, :, 2), a_cov(:, :, 1) + a_cov(:, :, 2));
    [D,ind] = sort(diag(D)); W = W(:,ind); W= W';

    % override CSP to check something
    load ('c:\csp.txt ');
    for ifil=1:ncat
        for i=1:ntrial(ifil)
            eeg_t(:, :, i, ifil)= W * eeg(:, :, i, ifil);
        end
    end

    tot_train= sum(x_train);
    tot_test= sum(x_test);

    % Classificatie loop
    for im=1:m_num
        n_train=0; n_test=0;
        train_feat= zeros(tot_train,2*im); test_feat= ↵
            zeros(tot_test,2*im);
        train_group= 0 * (1:tot_train);    test_group= 0 *↵
            (1:tot_test);
        for hand=1:ncat
            for trial=1:ntrial(hand)
                for in=1:im
                    f(in)= var(eeg_t(in, :, trial, hand));
                    f(2*im + 1 - in)= var(eeg_t(nchan + 1 -↵
                        in, :, trial, hand));
                end
            end
        end
    end

```

```

end
totvar= sum(f(1:2*im));
for in=1:2*im, logf(in)= log(f(in)/totvar); ←
end
if selection(trial,hand) == 1
    n_train= n_train + 1;
    train_feat(n_train,1:2*im)= logf(1:2*im ←
    );
    train_group(n_train)= hand;
else
    n_test= n_test + 1;
    test_feat(n_test,1:2*im)= logf(1:2*im);
    test_group(n_test)= hand;
end
end
end

data.X=train_feat';
data.y=train_group';
model = fld(data);
[class dfce] = linclass(test_feat', model);
part_done = (( (ifreq - 1) * 10 * m_num) + (krun-1) ←
    *m_num + im)/(10*nfreq*m_num);
singPro(part_done,[num2str(floor(part_done*100)) ' ←
    % done.'], 'Calculating Hit-Rate');
totcor(ifreq,im) = totcor(ifreq,im) + sum(class == ←
    test_group);
end % im-loop
end
totcor(ifreq,:)= totcor(ifreq,:)/sum(ntrial(:));
end % ifreq-loop
singPro;

vargout = selectCSPGui(nfreq,m_num,totcor);
chosen_freq= vargout(1);
chosen_num= vargout(2);

filters=sort([lpfreq hpfreq]);
filters(1)= filters(1) + chosen_freq - midfreq;
filters(2)= filters(2) + chosen_freq - midfreq;

[FilterB,FilterA]= bandfilt(sr,filters(1)-2,filters(1),filters ←
    (2),filters(2)+2,0.1,30);

nchan=size(CalibrationSignal{1,1},1);
nsamp=size(CalibrationSignal{1,1},2);
ncat=size(CalibrationSignal,1);
maxtrial=size(CalibrationSignal,2);
ntrial=zeros(1,ncat);

eeg=zeros(nchan,nsamp,maxtrial,ncat);
eeg_epoch = zeros(nchan,floor(sr/2),nsamp*2/sr, ncat);
new_ntrial= 0 * (1:ncat);
for hand=1:ncat
    for trial=1:maxtrial
        if (size(CalibrationSignal{hand,trial})) ~= 0
            ntrial(hand)=ntrial(hand)+1;
            for channel=1:nchan

```

```

eeg(channel,:,trial,hand) = filtfilt(FilterB, ←
    FilterA, CalibrationSignal{hand,trial})(←
    channel,:));
end
for i=1:1000
    % beg= floor(sr/2 + (i-1) * sr/4);
    % beg= floor(sr/4 + (i-1) * sr/4);
    % beg= floor(3 * sr/2 - 3 * sr/10 + (i-1) * sr←
    % /10);
    % fin= floor(beg + sr/2 - sr/10 - 1);
    % beg= floor(sr/10 + (i-1) * sr/10);
    % fin= floor(beg + sr/2 - 1);
    % fin= floor(beg + 8 * sr/10 - 1);
    ns2= floor(fin-beg + 1);
    if fin <= (nsamp - sr/10)
    % if i <= 1
        new_ntrial(hand)= new_ntrial(hand) + 1;
        eeg_epoch(:,1:ns2,new_ntrial(hand),hand)= ←
            eeg(:,beg:fin,trial,hand);
    else
        break;
    end
end
end
end
end

clear eeg
eeg= eeg_epoch;
clear eeg_epoch
ntrial= new_ntrial;

% one more correction for mean

for hand=1:ncat
    for j=1:ntrial(hand)
        for i=1:nchan
            eeg(i,:,j,hand)= eeg(i,:,j,hand) - mean(eeg(i,:,j,←
            hand));
        end
    end
end

%Cmean = zeros(ncat);
for hand=1:ncat
    for trial=1:ntrial(hand)
        E = eeg(:, :, trial, hand);
        tmpC = (E*E');
        C{hand}(:, :, trial) = tmpC./trace(tmpC);
    end
    Cmean{hand}=mean(C{hand},3);
end

Ccomposite=Cmean{1}+Cmean{2};
% Sort eigenvalues in descending order
[Ucomposite,Lambdacomposite] = eig(Ccomposite);
[Lambdacomposite,ind] = sort(diag(Lambdacomposite),'descend');
Ucomposite = Ucomposite(:,ind);

```



```

% Ramoser equation (3) - Whitening transform
P=sqrt(inv(diag(Lambdacomposite)))*Ucomposite';

[W,D] = eig(Cmean{2},Ccomposite); % Simultaneous diagonalization
[D,ind] = sort(diag(D)); W = W(:,ind); W= W';

P=pinv(W); % Common spatial patterns

% apply CSP
load ('c:\csp.txt ');

for hand=1:ncat
    for trial=1:ntrial(hand)
        eeg_t(:, :, trial, hand)= W * eeg(:, :, trial, hand);
    end
end
f=[];

% train full classifier

train_feat = zeros(sum(ntrial),chosen_num);
train_group= zeros(sum(ntrial),1);
tn=0;

f = zeros(2*chosen_num);
for hand=1:2
    for trial=1:ntrial(hand)
        for in=1:chosen_num
            f(in)= var(eeg_t(in, :, trial, hand));
            f(2*chosen_num + 1 - in)= var(eeg_t(nchan + 1 - in←
                , :, trial, hand));
        end
        totvar= sum(f(1:2*chosen_num));
        for in=1:2*chosen_num
            logf(in)= log(f(in)/totvar);
        end
        tn=tn+1;
        train_feat(tn,1:2*chosen_num)= logf(1:2*chosen_num);
        train_group(tn)= hand;
    end %i-loop
end %ifil loop

% now fill in the calculate values in the bci parameter ←
configuration.
if size(bci_Parameters) ~=0

    bci_Parameters.UsePatterns{1,1} = num2str(chosen_num);
    bci_Parameters.lpfreq{1,1} = num2str(lpfreq + chosen_freq ←
        midfreq);
    bci_Parameters.hpfreq{1,1} = num2str(hpfreq + chosen_freq ←
        midfreq);

    for i=1:size(W,1)
        for j=1:size(W,2)
            bci_Parameters.CSPFilterMatrix{i,j} = num2str(W(i,j←
                ));
        end
    end
end
for i=1:size(train_feat,1)

```

```

        for j=1:size(train_feat,2)
            bci_Parameters.TrainingFeatures{i,j} = num2str(←
                train_feat(i,j));
        end
    end
    for i=1:size(train_group,1)
        for j=1:size(train_group,2)
            bci_Parameters.TrainingGroup{i,j} = num2str(←
                train_group(i,j));
        end
    end
end

% save matrices
save ('c:\csp.txt','-ascii','-tabs','W')
save ('c:\tf.txt','-ascii','-tabs','train_feat')
save ('c:\tg.txt','-ascii','-tabs','train_group')

if (Plot)
    Patterns= chosen_num;
    chan_name= 'elec_positions.txt'; % names and ←
        positions
    [nam,pos1,pos2,pos3]= textread(chan_name,'%s %f %f %f',112)←
        ;

    % labels=bci.colheaders(Channels);
    totch=0;
    Channels=1:length(labels);
    usedlabels={};
    for i=1:length(Channels)
        for j=1:108
            if strcmpi(labels(i),nam(j))
                totch= totch+1;
                usedlabels{i}=char(nam(j));
                pos(i,1)= pos1(j);
                pos(i,2)= pos2(j);
                pos(i,3)= pos3(j);
                break;
            end
        end
    end
    usedlabels=usedlabels';
    xx = pos(:,1); yy = pos(:,2); zz = pos(:,3);
    cfg.interplimits= 'head';
    cfg.emarkersize= 4;
    cfg.colorbar= 'no';
    cfg.electrodes='labels';
    cfg.efontsize =7;
    blocksize=BlockSize*(HistoryBlocks - 3);
    nn=floor(blocksize/2) + 1;
    pow= zeros(nn,2*Patterns,2); % to store spectra
    for comp=1:Patterns
        for hand=1:2
            for trial=1:ntrial(hand)
                [sp,freqs]= pwelch(eeg_t(←
                    comp,:,trial,←
                    hand),blocksize,0,blocksize,sr);
                pow(:,comp,hand)= pow(:,comp,hand) + sp(:);
                [sp,freqs]= pwelch(eeg_t(end+1-comp,:,trial,←
                    hand),blocksize,0,blocksize,sr);
            end
        end
    end
end

```

```

        pow(:,end+1-comp,hand)= pow(:,end+1-comp,hand) ←
            + sp(:);
    end
    pow(:, :, hand)= pow(:, :, hand)/ntrial(hand);
end
end

hold off;
freqlrange=[1 30]; %filterb.*[.8 1.3];

for component=1:Patterns
    subplot(Patterns,4,4*(component-1)+1);
    for i=1:2
        if i==1, x= plot(freqlrange, pow(:, component, i), 'b'); ←
            end
        if i==2, x= plot(freqlrange, pow(:, component, i), 'r'); ←
            end
        set(gca, 'xlim', freqlrange);
        hold on
    end
    ylabel('Power');
    xlabel('Frequency (Hz)');
    plot([filters(1) filters(1)], ylim);
    plot([filters(2) filters(2)], ylim);
    Title=sprintf('Component : %i of CSP', component);
    title(Title);
    hold off
    subplot(Patterns,4,4*(component-1)+2);
    for i=1:2
        if i==1, x= plot(freqlrange, pow(:, end+1-component, i), ←
            'b'); end
        if i==2, x= plot(freqlrange, pow(:, end+1-component, i), ←
            'r'); end
        set(gca, 'xlim', freqlrange);
        hold on
    end
    ylabel('Power');
    xlabel('Frequency (Hz)');
    plot([filters(1) filters(1)], ylim);
    plot([filters(2) filters(2)], ylim);
    Title=sprintf('Component : %i of CSP', component);
    title(Title);
    subplot(Patterns,4,4*(component-1)+3);
    cfg.colorbar= 'no';
    topoplot(cfg, xx,yy,W(component,:),usedlabels);
    % topoplot(cfg, xx,yy,P(:,component),usedlabels);
    title(['CSP ' int2str(component)]);
    subplot(Patterns,4,4*(component-1)+4);
    cfg.colorbar= 'no';
    % topoplot(cfg, xx,yy,W(component,:),usedlabels);
    topoplot(cfg, xx,yy,W(nchan + 1 - component,:), ←
        usedlabels);
    % topoplot(cfg, xx,yy,P(:,nchan + 1 - component), ←
        usedlabels);
    % title(['Filter ' int2str(component)]);
    title(['CSP ' int2str(component)]);
    hold off
end
end
end

```

```
end
```
